Measuring the quality of publicly available synthetic IDS datasets

A comparative study

A. Brussen

Measuring the quality of publicly available synthetic IDS datasets

A comparative study

by

A. Brussen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday August 19, 2021 at 1:30 PM.

Student number:	4974484	
Project duration:	September 9, 2020 – August 19, 2021	
Thesis committee:	Dr. ir. S.E. Verwer,	TU Delft, supervisor
	Dr. ir. J.H. Krijthe,	TU Delft
	R. Brouwer,	Deloitte Cyber Risk

Abstract

Year after year, the amount of network intrusions and costs associated to them rises. Research in this area is, therefore, of high importance and provides valuable insight in how to prevent or counteract intrusions. Machine learning algorithms seem to be a promising answer for automated network intrusion detection, as their results often reach upwards of 99%+ on datasets. Yet even with these results, the problem does not seem to be solved as the same models do not reach similar scores on real live network traffic. This indicates a problem with the datasets.

In this work, we explored six recent network intrusion datasets and measured the quality of them from a design perspective and through a practical binary classification approach. Furthermore, we explored the need for complex classification models on these datasets, as research has shifted more towards using black-box models as opposed to white-box models.

Through a literature study on the quality metrics of a dataset, we found there is a general lack of agreement amongst researchers regarding what makes a dataset good and realistic. We also found areas in which datasets are often lacking in and provide concrete advice on how to upgrade the quality of these datasets.

For our practical classification approach, we built a general classification pipeline using a Random Forest. Three feature sets were tested, two of which form an ablation study to measure the effects of trigrams on the classification score. On all datasets the general classification model reaches good classification results (>90%+) and for three datasets even reached state-of-the-art results. The ablation study yielded a positive effect of using trigrams for classification on the datasets. Our white-box approach performed on par or better than most black-box techniques. We conclude that black-box models are unnecessary in this problem context and that the techniques should shift back to white-box approaches.

Next, we attempted to link the quality of dataset methodologies to the difficulty of obtaining state-of-the-art classification results. Apart from the complexity of the attack vector and benign traffic variety, we found no further properties that define this relationship.

Finally, this work started with the assumption that real live network traffic is more complex and therefore more difficult to classify well on than most available datasets. Newer datasets do show improvements over older datasets and our classification results corroborated the validity of the assumption.

Preface

What a time to be alive, writing a thesis during a pandemic. Can't really recommend it, to be honest. I could write another thesis on the effects of a pandemic on students writing their master's thesis, but let's just not.

I would like to thank my supervisor for helping me find a research topic I'm interested in. I am not sure I would have succeeded without the ideas and feedback I got that steered me in this direction. Also more generally, I would like to thank him and the rest of the research group for the weekly meetings and game nights in the Gather environment. It definitely helped with the social aspect during the lockdowns!

Next, I would like to thank Deloitte for the opportunity to do my thesis with them. Especially my main supervisor Rayan and my secondary supervisor Daan, thanks for being there so often - it meant a lot! The same goes for my thesis coordinator Sabien and my fellow thesis interns there; thanks for the pep talks and I hope to see you more often!

Finally, big thanks to my family and friends for being the distraction I sometimes heavily needed, however big or small. I couldn't do it without you.

A. Brussen
Delft, August 2021

Contents

1 Introduction

Over the years, the use of computers has evolved our lives and low-latency global communication has become more prevalent than ever. From simple desktop computers and landlines to dynamically-scaled cloud computing architectures and smartphones, technology is advancing at an increasing rate. With the increasing popularity also comes a massive increase in the amount of traffic that is being generated and processed. Combined with the increasing complexity of network structures, securing such networks has become much more time-consuming and difficult over the years.

Network administrators are fighting a continuous battle against malicious threat actors, whose goal are to acquire sensitive information or to attack the availability of services. In 2019, the average cost of a data breach was $3.92 million[1]. Moreover, the average cost of a malware attack on a company is $2.6 million[2]. Properly configuring and protecting all network assets is vital in preventing unwanted access to internal systems or databases. It is, however, naive to assume that network configuration alone is enough to defend against ill-intended threat actors. Therefore, it is imperative that security is applied in multiple layers, for instance by also monitoring network traffic.

The sheer volume of network traffic within any reasonably-sized network is infeasible to be fully analysed by hand. Automated analysis is a much-needed tool to filter out benign traffic from malicious traffic. A Security Operations Center (SOC) will, with the help of an intrusion detection system (IDS), focus on the traffic being flagged as malicious. The classification quality of the IDS greatly affects the amount of manual labour done by the SOC. Due to the enormous class imbalance of benign vs. malicious traffic in datasets, accuracy is not the most important metric that should be looked at. The amount of false positives, i.e. the amount of benign traffic classified as malicious, has an important effect on the inspection quality of SOC analysts. Too many false positives and the SOC analysts might incorrectly flag a true positive as false positive as well. The amount of false negatives on the other hand, i.e. the amount of malicious traffic classified as benign, undermines the foundation of such a detection system, as the SOC analysts are not even alerted of the malicious traffic; attacks could be completely invisible for the IDS.

1.1. Motivation

The last ten to fifteen years, more focus has been set on researching countermeasures against unwanted network intrusion. Signature-based, anomaly-based and DNS-based machine learning are all detection methods that have been researched for this problem [22]. Even for recent datasets, simple machine learning methods perform rather well on datasets. For instance, [32] contributed a dataset called CICIDS2017 that meets the necessary criteria designed by a paper from 2016 [11]. with additional common attacks. It is a network trace that ran for 5 days, in which benign activity and different attacks took place. Furthermore, they performed a feature analysis using Random Forest Regression to construct the best feature subset (4 features in a subset) per attack. The analysis shows that each type of attack has its own distinct feature set. With these best feature subsets in mind, they tried seven general machine learning techniques for classification. Figure 1.1 shows the results. A simple

[1]`https://securityintelligence.com/posts/whats-new-in-the-2019-cost-of-a-data-breach-report/`
[2]`https://www.accenture.com/us-en/insights/security/cost-cybercrime-study`

clustering method, k-NN, has a classification F1-score of 0.96, indicating that there exists a simple classification technique which can accurately separate malicious from benign traffic.

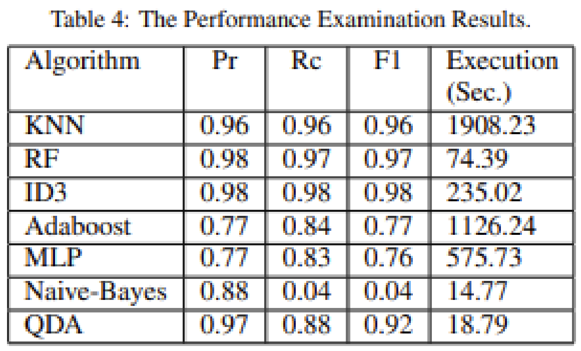


Table 4: The Performance Examination Results.

Algorithm	Pr	Rc	F1	Execution (Sec.)
KNN	0.96	0.96	0.96	1908.23
RF	0.98	0.97	0.97	74.39
ID3	0.98	0.98	0.98	235.02
Adaboost	0.77	0.84	0.77	1126.24
MLP	0.77	0.83	0.76	575.73
Naive-Bayes	0.88	0.04	0.04	14.77
QDA	0.97	0.88	0.92	18.79

Figure 1.1: Results of several ML techniques on anomaly detection of several attacks

A more advanced white-box technique, random forest, performs even better with its main strength being the significantly lower execution time. Obtaining high classification scores with white-box techniques is great for obtaining insights into patterns that separate the data in distinct groups, as it allows researchers and analysts to learn from the decision process, which is especially true for implementations such as decision trees and random forests. The fact that a high percentage of data can be explained using white-box machine learning methods tells us one of two things:

- The problem of separating malicious from benign traffic is not as complex as thought to be;
- The complexity of this public dataset (this translates to nearly all other public datasets as well!) is insufficient for state-of-the-art anomaly detection research.

The first statement can be refuted simply by inspecting the increasing amount and severity of cyber attacks on organizations. High-profile organizations are still subjected to attacks and suffer the lost confidentially or integrity of their crown jewels. It is, therefore, short-sighted to state the problem of intrusion detection as solved based on the classification results of (publicly available) IDS datasets. It is more likely that the datasets themselves are easy to correctly classify and that the used evaluation metrics, which often only is accuracy or F1-score, are insufficient to fully describe the results.

In a previous master thesis by Serentellos [30], an exploratory analysis on the CTU-13 intrusion detection dataset revealed that simply classifying data on source port gave near perfect classification results. The infrastructure included Windows XP SP2 operating systems for the victim virtual machines running on top of a Linux Debian host. Each virtual machine was bridged into the university network, which gathered normal and background traffic used in the final dataset. The Linux Debian machine exclusively generated botnet behaviour and was used for labelling purposes. The differences between Windows and Linux operating systems cause both architectures to choose varying port numbers and TCP/IP [2] properties in network traffic. A classification model, therefore, heavily prefers using source ports as a feature, even though a combination of other features such as protocol, packet size in bytes, flag values may work just as well and generalize to other infrastructure. Instead, the found solution is very specific to this dataset and provides little insight about network intrusion detection in a broader context.

Over the years, multiple papers analysed the quality of intrusion detection datasets [11, 12, 14, 28, 32, 34]. Often, these papers bring along or result in new datasets that cover all the proposed properties that should be in a high-quality dataset, according to the authors of each paper. We believe, however, that most public datasets, whether they meet those properties or not, are not representative for real traffic data and are thus unsuitable for state-of-the-art research. General traffic patterns found in one dataset are not necessarily found in other similar datasets, illustrating the unstandardised methodology of constructing research-focused network traffic datasets.

With the increase in popularity of (deep) neural networks, these have also become much more prevalent in recent anomaly detection journals and conferences [5, 24, 40, 43]. From a research perspective, this is peculiar, as traditional machine learning methods tend to work better on structured data, whereas neural networks work better with unstructured data.

Compared to traditional methods, e.g. k-NN clustering or decision trees, the insight gained from papers utilizing neural network implementations is minimal due to the black-box nature of neural networks. We suspect that, due to network traffic data being structured and good classification results by prior work [1, 10, 13, 19, 41], (supervised) white-box techniques provide more insight into possible patterns of the data, compared to black-box techniques, while maintaining similar classification scores.

1.2. Problem Statement

The problem that will be tackled in this thesis concerns the quality of synthetic IDS datasets and the complexity of corresponding state-of-the-art anomaly detection solutions. We define the quality of an IDS dataset by the representativeness of its events. In other words, compared to live traffic data in the same domain or industry, how realistic are the events in the IDS dataset? Researchers are constantly finding new ways to improve classification scores above 99% on these datasets, yet their models do not translate well to live environments. This indicates a problem with the quality of the datasets used. We work on the assumption that realistic traffic is more complex and thus more difficult to classify well on, as classification results on these datasets are excellent yet intrusion detections within real organisations happen more frequently every year.

The measure of quality described by [14] will form the foundation for this. Figure 1.2 provides an overview of qualitative properties datasets should have. Of large importance is the property about event relevance. Often, event patterns are not realistic, as they are not found in other datasets or live data in the same domain. Dataset provenance, i.e. clarity of the methodology and metadata description, is also a critical factor in measuring the quality of datasets, as researchers often work with incomplete information and sometimes have to draw their own assumptions. Finally, we see domain context as another important indicator of quality, as each domain has wildly different traffic patterns and threat compositions.

Next, we wish to study the need for complex models in this field. We define the complexity of a model by the interpretability of the results; can a SOC analyst understand how the anomaly detection solution made its decision? In the case of a decision tree implementation, it is trivial to see how the solution made its choice. When using something more complex, for instance (deep) neural network implementation, the sheer amount of interacting nodes and raw weight values obfuscate the reasoning behind the decision. Needless to say, if a less complex classification model achieves the same classification score as a more complex method, then in the same vein as Occam's Razor, the usefulness of the simpler model is higher.

1.3. Research Questions

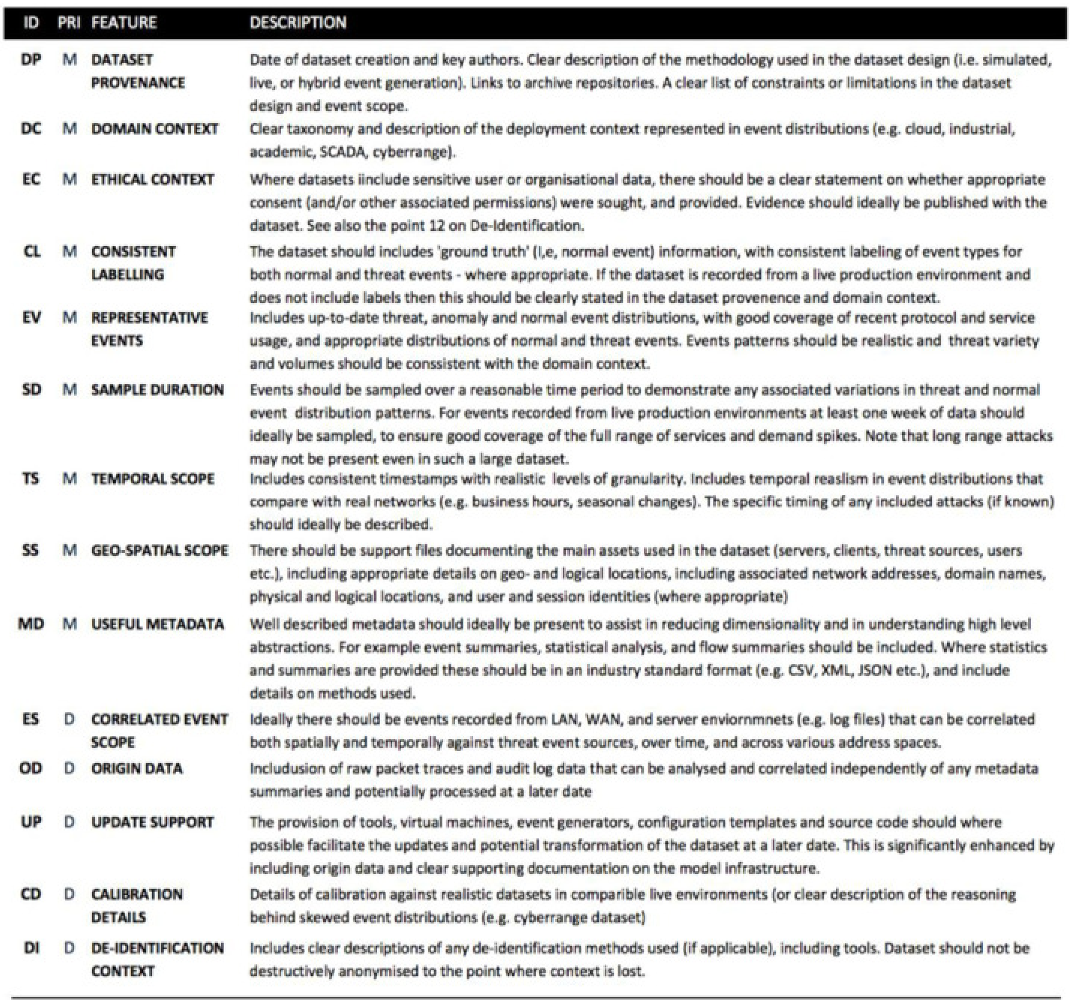
Dataset quality measurements are often done in comparative studies to expose potential issues with existing datasets. Often, these datasets have been generated using metrics designed by the authors or by referenced papers. A literature study on these comparative studies and the designed metrics is interesting, as it allows us to draw conclusions on the perceived quality of datasets and the methodology behind them from a design perspective. Our first research question will focus on the observations that we can make by studying the used quality metrics of creating IDS datasets:

- Are the sets of quality metrics used by dataset creators suitable for creating realistic datasets?
  - Is there consistency in the use of metrics and quantification of quality over time?
  - What quality metrics are considered more important than others?

We hypothesize that, since there is no general consensus on designing IDS datasets, that there will be inconsistencies in defining the properties and metrics that the datasets should have. Moreover, we suspect these issues to stem from a lack of knowledge and thorough analysis.

Furthermore, we are interested in testing the quality of existing public datasets and complexity of corresponding solutions by means of creating a general classification model. With those results, we would like to see if there is relationship between the design quality of a dataset and the classification results. This brings us to our second research question:

- Is there a relationship between the design quality of a dataset and the difficulty of obtaining state-of-the-art classification results?



ID	PRI	FEATURE	DESCRIPTION
DP	M	DATASET PROVENANCE	Date of dataset creation and key authors. Clear description of the methodology used in the dataset design (i.e. simulated, live, or hybrid event generation). Links to archive repositories. A clear list of constraints or limitations in the dataset design and event scope.
DC	M	DOMAIN CONTEXT	Clear taxonomy and description of the deployment context represented in event distributions (e.g. cloud, industrial, academic, SCADA, cyberrange).
EC	M	ETHICAL CONTEXT	Where datasets iinclude sensitive user or organisational data, there should be a clear statement on whether appropriate consent (and/or other associated permissions) were sought, and provided. Evidence should ideally be published with the dataset. See also the point 12 on De-Identification.
CL	M	CONSISTENT LABELLING	The dataset should includes 'ground truth' (I,e, normal event) information, with consistent labeling of event types for both normal and threat events - where appropriate. If the dataset is recorded from a live production environment and does not include labels then this should be clearly stated in the dataset provenence and domain context.
EV	M	REPRESENTATIVE EVENTS	Includes up-to-date threat, anomaly and normal event distributions, with good coverage of recent protocol and service usage, and appropriate distributions of normal and threat events. Events patterns should be realistic and threat variety and volumes should be conssistent with the domain context.
SD	M	SAMPLE DURATION	Events should be sampled over a reasonable time period to demonstrate any associated variations in threat and normal event distribution patterns. For events recorded from live production environments at least one week of data should ideally be sampled, to ensure good coverage of the full range of services and demand spikes. Note that long range attacks may not be present even in such a large dataset.
TS	M	TEMPORAL SCOPE	Includes consistent timestamps with realistic levels of granularity. Includes temporal reaslism in event distributions that compare with real networks (e.g. business hours, seasonal changes). The specific timing of any included attacks (if known) should ideally be described.
SS	M	GEO-SPATIAL SCOPE	There should be support files documenting the main assets used in the dataset (servers, clients, threat sources, users etc.), including appropriate details on geo- and logical locations, including associated network addresses, domain names, physical and logical locations, and user and session identities (where appropriate)
MD	M	USEFUL METADATA	Well described metadata should ideally be present to assist in reducing dimensionality and in understanding high level abstractions. For example event summaries, statistical analysis, and flow summaries should be included. Where statistics and summaries are provided these should be in an industry standard format (e.g. CSV, XML, JSON etc.), and include details on methods used.
ES	D	CORRELATED EVENT SCOPE	Ideally there should be events recorded from LAN, WAN, and server enviornmnets (e.g. log files) that can be correlated both spatially and temporally against threat event sources, over time, and across various address spaces.
OD	D	ORIGIN DATA	Includusion of raw packet traces and audit log data that can be analysed and correlated independently of any metadata summaries and potentially processed at a later date
UP	D	UPDATE SUPPORT	The provision of tools, virtual machines, event generators, configuration templates and source code should where possible facilitate the updates and potential transformation of the dataset at a later date. This is significantly enhanced by including origin data and clear supporting documentation on the model infrastructure.
CD	D	CALIBRATION DETAILS	Details of calibration against realistic datasets in comparible live environments (or clear description of the reasoning behind skewed event distributions (e.g. cyberrange dataset)
DI	D	DE-IDENTIFICATION CONTEXT	Includes clear descriptions of any de-identification methods used (if applicable), including tools. Dataset should not be destructively anonymised to the point where context is lost.

Figure 1.2: Identified qualitative properties that researchers should consider when constructing new intrusion detection datasets [14]. M = Mandatory, D = Desirable.

- Is it possible to create a general white-box classification model that works well on multiple network intrusion datasets?
- Are state-of-the-art solutions by other researchers overcomplicated, i.e. are our results similar or possibly even better than theirs?

By answering these sub questions, we find out whether there exist common features that work well on multiple datasets. Also, a comparison with related work on the classification scores reveals the necessity of complex classification models on network intrusion detection. We expect there to exist some common features between datasets, although the importance of each feature may differ due to attack diversity and traffic distribution. We also believe to achieve good classification scores ($> 90\%$) with the general classification model, although we do not anticipate similar scores as from related work on every dataset. With the answers to these subquestions, we can answer the second research question together with our assumption that realistic data is more difficult to classify well on than unrealistic data. We suspect, due to high classification results by related work and our own expected classification results, that most public datasets are unrealistic and that we can find a relationship between the design quality and the difficulty of obtaining great classification results.

1.4. Proposed Solution

The aim of this study is to determine the quality of existing synthetic datasets and corresponding solutions. Our investigation is two-fold. First, we will perform a literature study on the design/evaluation metrics of datasets over the years. The goal is to find out how the perceived importance of certain metrics have changed over the years and why there is still a lack of consistency between dataset creators. Next, we can inspect the methodology of the datasets we will use with our findings. From this study, we can draw conclusions on the developments over the years, the current state and potential improvements in the design methodology of IDS datasets.

The second angle will consist of a comparative study that judges the quality of datasets from a practical point of view. To study the need for complex models in classifying anomalous behaviour, we will design a white-box and, therefore, interpretable classification method that we can apply on the selected datasets. In general, we will perform this experiment on flow-based (connection-level) network intrusion datasets, as those are most prevalent, up-to-date and well-labelled. In the case that one of the flow datasets also carries separate packet data which can be labelled using the flow-based data, we will use the packet-level data as well. To facilitate the generalization of this model, we design a pipeline process through which each dataset can pass and output classification results. The pipeline will consist of preprocessing the datasets to a usable form, selecting features from each dataset for the generalized classification model, optimize said model and finally report useful metrics per dataset to compare with related work. Depending on the quality and quantity of common features, and classification results compared to prior work, we can draw conclusions on the quality of the datasets and complexity of models from a practical perspective. With that information, we can return to our claim in Section 1.1 about the peculiarity of obtaining amazing results that do not translate well to live environments and link it to our results.

1.5. Contributions

We present a list that summarises the contributions of this work:

- We reflect on useful evaluation metrics of classification methods within the context of network traffic anomaly detection. We show that accuracy, an often used quality metric, does not work with the inherent class imbalance of this problem, and that classification works should always provide a confusion matrix with the summarised classified instances;
- An overview of available works that assess the quality of IDS datasets;
- A list of dataset quality metrics that are commonly overseen or underappreciated by researchers and ways to incorporate those metrics in the design methodology;
- A comparison between state-of-the-art classification models and our classification models on multiple datasets, wherein we illustrate the similar results between the models and, therefore, the unnecessary complexity of some models;
- An ablation study focussing on the usefulness of trigrams in network intrusion detection, showing a positive effect of using trigrams when constructing the feature set.

1.6. Document Structure

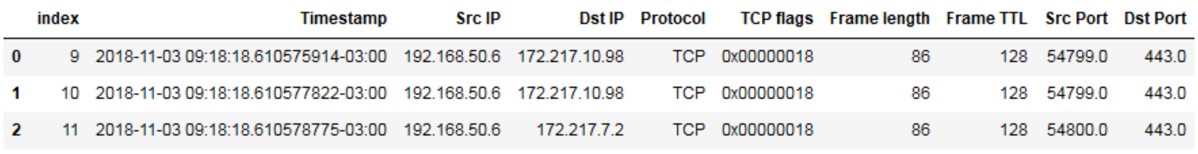
First, we provide a background of relevant knowledge to ensure the reader understands the necessary concepts. Chapter 3 then presents the related work relevant for RQ1 and the analysis thereon. The chapter is concluded with a conclusion and corresponding contributions. The next chapter concerns the classification pipeline and starts with a list of data requirements and chosen datasets, each with a summary of their data composition. Then, a related works section on papers which reflect the current state of network traffic anomaly detection is shown along with relevant techniques that we will explore. Beyond that, we describe each component of the pipeline, experiment setup and results thereof. Finally, we discuss potential improvements and future work, reflect on the answers to the research questions and conclude this work with a short summary of what has been done.

2 Background

In this chapter we will provide the necessary knowledge to understand the rest of this work. In particular, we will explain what network traffic is and how it can be represented, its sequential property, what intrusion detection entails and what performance metrics are important in this field.

2.1. Network Traffic Data

With the never-ending growth of the field of Information and Communications Technology (ICT), it is important to understand the basic building blocks of network communication. A computer network forms the primary structure of the digital landscape, and is connected to other computer networks through the internet. Network traffic in its basic form consists of streams of network packets from a source device to one or multiple destination device(s). A network packet is a structured unit of data containing control information (the header) and potentially user data (the payload). The header is responsible for providing the information needed for the correct transmission of packets. Along with routing information, it also contains error detection codes and possibly sequencing information. The payload provides the actual data that is to be inspected and possibly used by the receiving party. By using the appropriate protocols, the user can send a packet from a local device to another device outside their local network, given that both devices are connected to the internet. An example of a packet can be seen in Figure 2.1. (Inter)network communication by design is not secure, and requires additional security tools to prevent malicious transmissions. For large organisations, this poses to be a big challenge due to the many malicious actors interested in breaking the confidentiality, integrity or availability (CIA) of their services or data. Therefore, they invest substantial resources to combat and monitor malicious network traffic.



	index	Timestamp	Src IP	Dst IP	Protocol	TCP flags	Frame length	Frame TTL	Src Port	Dst Port
0	9	2018-11-03 09:18:18.610575914-03:00	192.168.50.6	172.217.10.98	TCP	0x00000018	86	128	54799.0	443.0
1	10	2018-11-03 09:18:18.610577822-03:00	192.168.50.6	172.217.10.98	TCP	0x00000018	86	128	54799.0	443.0
2	11	2018-11-03 09:18:18.610578775-03:00	192.168.50.6	172.217.7.2	TCP	0x00000018	86	128	54800.0	443.0

Figure 2.1: Small sample of packet-level data in a dataset. Only includes specific information parsed with *tshark*.

2.1.1. Monitoring and Analysis

A security operations centre (SOC) is a centralized unit within an organisation that deals with security issues. For the field of I(C)T, the official terminology is an information security operations centre (ISOC). Their primary activities include monitoring, analysis, control and removal of threats. Monitoring and analysing every inbound and outbound packet in a network is becoming less feasible every year, as the amount of users, web services and general digital communication grows. However, with that growth also comes the growth in expertise regarding monitoring.

Intrusion detection is often done with (one of) two types of data. Host-based intrusion detection systems (HIDS) analyse event logs from devices to find protocol violations or anomalous instructions.

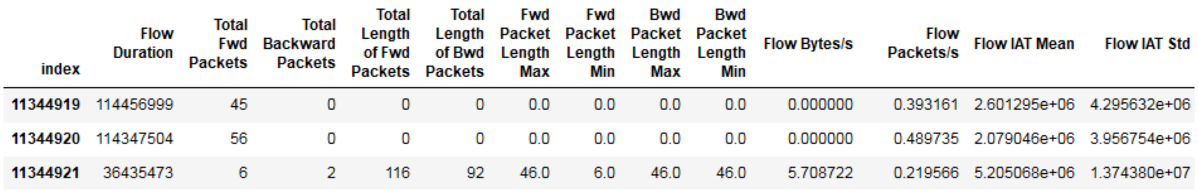
Network-based intrusion detection systems (NIDS), on the other hand, utilize network traffic data to detect malicious behaviour. Over the years, various network monitoring techniques have been proposed and implemented in monitoring tools.

Monitoring network traffic can be done in two ways: active and passive. Active monitoring is a proactive method where packets are injected into the network and tracked through monitoring tools. The idea is to analyse the behaviour of services as they react to the packets. Even though the method introduces additional traffic onto the network, the extra traffic is often fully customizable and proves to be effective even in small amounts. As it is equal to a controlled experiment, it is ideal for measuring specific metrics or behaviours. Conversely, passive monitoring does not inject data into the network, but monitors the existing traffic streams. It allows for a holistic overview of a network's behaviour and an analysis on the network's current health. It is used to gather large quantities of data, to be analysed at a later point in time, similar to an observational study. Due to the amount of data it gathers, it may require specialized hardware to measure and store all the traffic. The datasets used in this work are acquired through passive monitoring.

Some organisations are forced to sample their network traffic, because the raw transmission rate is too high for their monitoring hardware to handle. This generally is undesirable, as sampling causes a proportion of the data not to be analysed. Monitoring and analysis can be done on three levels: Packet-level, connection-level, and host-level. Packet-level allow per-packet monitoring and analysis as can be inspected in nearly all network traffic inspection programs, such as Wireshark[1]. Connection-level drops the packet payloads and groups the packets to their respective network flow representations, often defined by the 5-tuple:

- Source/destination IP addresses;
- Source/destination ports (if any);
- Protocol used for communication.

An example is given in Figure 2.2. The primary advantage is found in the reduced data size. For most purposes, deep-packet inspection, where the payload contents are analysed, is unnecessary for effective network monitoring. Moreover, it is a valid assumption that if one packet is malicious, then the flow of packets that correspond with it may carry malicious patterns or information, as well. Another advantage of excluding the payload content in the data is the increase in privacy of the data. Most organisations do not have the resources to maintain an in-house SOC and analyst team, therefore a third party is contracted to analyse the flow-based data. Due to these reasons, nearly all available public intrusion detection datasets already have flow-based versions.



index	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Bwd Packet Length Max	Bwd Packet Length Min	Flow Bytes/s	Flow Packets/s	Flow IAT Mean	Flow IAT Std
11344919	114456999	45	0	0	0	0.0	0.0	0.0	0.0	0.000000	0.393161	2.601295e+06	4.295632e+06
11344920	114347504	56	0	0	0	0.0	0.0	0.0	0.0	0.000000	0.489735	2.079046e+06	3.956754e+06
11344921	36435473	6	2	116	92	46.0	6.0	46.0	46.0	5.708722	0.219566	5.205068e+06	1.374380e+07

Figure 2.2: Small sample of flow-based data in a dataset. Some features, including protocol value and class label are omitted due to display size.

Finally, host-level monitoring assigns classification decisions based on the source of each data point, i.e. the source IP is classified malicious or benign based on its behaviour in network traces. The advantage of this builds on the assumption that a host who sends out malicious traffic will most likely send more malicious traffic in the near future. By aggregating the classification results by source IP and by setting a threshold for when a device is considered malicious, a clear distinction between malicious and benign hosts will appear, providing high-level intelligence that can be acted upon. On the other hand, the need for a threshold also brings a significant disadvantage: In realistic networks, network devices often run several services and communication streams concurrently, most of which are not malicious. Infected devices, which sometimes contact other devices in the network or attempt to communicate with outside IP addresses, may also have perfectly legitimate communication streams

[1]`https://www.wireshark.org/` Retrieved on 14 January 2021

open at the same time. Aggregating data this far may cause significant accuracy loss in monitoring and analysing, as information is only available on a per-host basis.

2.1.2. Sequential Property

An important property of network traffic is sequentiality. When devices communicate with each other through protocols, they follow a (mostly) deterministic order of sending and receiving packets. This can be found on many levels on the protocol stack, for instance in TCP communication with corresponding acknowledgements, or HTTP(S) GET-requests and respective download of data. For traffic analysis, this has great implications as packets between two hosts can be mapped sequentially to provide an overview of the communication stream, allowing effective high-level analysis. Applications of this can be found in, for example, research concerning (behavioural) fingerprinting for network traffic. Besides sequentiality appearing in intra-packet patterns, it also appears in network flows. Due to hardware or operating system details, each device on a network has its own micropatterns. These micropatterns may differ per device and per reboot, but occur consistently in a single session. An interesting technique to make use of the sequential property within network data is by utilizing N-grams. Mostly used in the field of computation linguistics and probability, N-grams work by building a contiguous sequence of n items from a dataset, which potentially provides more information than the items by themselves. In the context of network data, that can be applied by aggregating data instances by some identifier, such as matching packets to their communication flow, or netflows to the source host. Then, a sliding window could be applied that calculates interesting characteristics over the n data instances for each valid row in the sliding window.

For offline analysis, large datasets are used to ensure enough data is available to draw meaningful conclusions. Properly sampling the data for training, testing and validation purposes is not a straightforward task, due to these (micro)patterns and sequential property. If a time window is used, it is important to choose the right length of the window, as a sliding window that is too small may not hold enough information and meet the sequential property of traffic data to be useful, but a sliding window that is too big may include too much or irrelevant information and distort results. In general, it is best to split the data based on timeframes. For instance, if a dataset is recorded on three separate days and all classes appear in every trace, then one day could suffice for training, and the other days for testing and/or validation. This ensures that the sequential property and, to some extent, the micropatterns are respected to reduce potential overfitting.

2.2. Network Intrusion Detection

Network IDS (NIDS) implementations often utilize one out of two major techniques: Signature-based detection and Anomaly-based detection. Both techniques have their advantages and drawbacks. Each NIDS monitors incoming and outgoing network traffic for anomalous behaviour.

2.2.1. Signature Detection

Signature-based detection is used widely and with great success to detect known attacks. For each instance it evaluates, a large knowledge base, filled with signatures of anomalous behaviours, is consulted. If the instance signature matches a known signature, then the instance is classified as anomalous, otherwise it is classified as benign. Besides building and maintaining a proper knowledge base, search optimization is also a major challenge in creating a signature-based IDS. If the lookup time for a single instance takes too long, then it will bottleneck the network bandwidth. One very popular field for signature detection is antivirus detection. Nearly all antivirus solutions use this technique for their detection algorithm.

Although relatively simple to design, the main drawback in these kinds of system is that zero-day attacks, i.e. unknown attacks, are not detected and simply pass through the IDS. For antivirus solutions, this drawback is abused by malware-writers by simply recompiling binaries with some randomization. Depending on how the solution defines a signature, it could completely bypass the IDS and therefore infect the target device with minimal extra effort. Likewise, for NIDS, randomizing packet information, fragmenting packets, changing the order of packets can all result in different signatures, depending on the implementation of the NIDS.

2.2.2. Anomaly Detection

Where signature detection compares traffic against known malicious signatures, anomaly detection compares traffic against baseline behaviour of a network, i.e. how benign traffic should look like. Traffic that does not conform to that baseline will be classified as anomalous. The design and construction of such a baseline can be done in various degrees. Having access to a priori information about the network significantly restricts the set of expected traffic in a network, therefore making the problem easier. An interesting and important application of this can often be found in NIDS for industrial control systems (ICS). Using design specifications and other knowledge, restricted sets of traffic fingerprints can be built to define normal behaviour. The result is often highly-effective as it is specifically tailored for that network.

However, not every network or domain has access to a priori information. Moreover, updates to a network, for example by updating the firmware of devices or adding new components, may change the baseline behaviour, causing the a priori information to be invalid for the new setup. Every update requires the NIDS to retrain, but also to re-evaluate the a priori information. Besides, such a solution does not carry over well to other similar setups, as the model is highly specific to one environment. As a result, most research into network-based anomaly detection does not use a priori information, but operates on the assumption that the baseline can be fully learned by training on enough representative data.

Training an anomaly detection system using only a dataset containing network traffic is no easy task. Depending on the domain, there could be many different services, workstations and other components communicating with each other and the internet. Traffic anomalies may have nearly the same patterns as normal, benign traffic. Therefore, an in-depth analysis on the network traffic data is needed to expose any patterns that differentiate normal traffic from anomalous traffic. If the patterns form a relatively simple baseline, then the NIDS could consist of a set of filtering rules, e.g. with profiling. Often, however, obtaining clear patterns is not a trivial task and sometimes the variation in data makes it too complex to design a proper baseline. Using machine learning, more powerful techniques could be used to find and exploit patterns for getting better results. For instance, a pattern that separates benign traffic from malicious traffic might exist in a high-dimensional space and consist of multiple clusters. Simple rule filtering or binning would most likely be ineffective or very difficult to maintain, as the classification has to be done based on multiple specific value ranges due to the high dimensionality. Instead, a clustering algorithm such as k-NN or a statistical (non-linear) algorithm such as a Support Vector Machine carry much better odds of finding these patterns.

2.2.3. Real vs. Synthetic Datasets

A very important distinction between intrusion detection system datasets is how the data was acquired. In this work, we make the distinction between real and synthetic datasets. Moreover, data from both sources can be combined by, for instance, adding artificial attacks to real captured data.

A real dataset only contains live captured data from an organization, which can then be analysed for behavioural patterns. To capture enough data for establishing a behavioural baseline, the length of real datasets often exceed a full workday. The obvious upside to real datasets is that they contain representative events, with respect to the domain and workflows of the organization. Building an IDS around this data, therefore, is a very effective solution for the organization to prevent unexpected traffic patterns. However, it often does not generalize well to other domains or similar organizations, as they have different workflows or software solutions that do not adhere to the defined baseline of the first organization's data. Moreover, and arguably also the largest issue, is the lack of privacy with real data. Since the data contains highly specific patterns and data distributions, it essentially maps all behaviour necessary to understand the infrastructure. Although the knowledge on data anonymization is improving over the years, it does destroy potentially useful information for building an IDS, and organizations are still very hesitant to share anonymized data due to the work needed to process it correctly and the risk involved of leaking infrastructure intelligence.

To tackle both these issues, synthetic datasets can be *created*. These datasets do not contain live captured data from real organizations, but instead consist of simulated data. It is very flexible, as the creators have complete control over every aspect of the dataset, including the infrastructure, threat composition and general event patterns. Because of this, privacy is also a much lesser issue, and instead the focus is on the correct methodology of the dataset. The complete control and flexibility of a synthetic dataset means that it is a very time consuming process to create a (correct) dataset.

Furthermore, it is difficult to verify its quality, as there exist many different definitions of what constitutes a good dataset.

Injecting, also known as mixing or salting, of malicious events in real benign data is sometimes also done if the original dataset does not contain a sufficient proportion of attacks or anomalous behaviour. However, it is considered frowned upon, as the datasets come from different sources. The data could then become unrelated, where the artificially added data differs significantly from the original data and thus does not represent relevant threat events within the original dataset.

2.3. Classification Methods

The goal of classification is to predict the type or class of an instance based on available properties or features. A trivial example is to have instances of animals, and to predict whether the animal is a dog, bird or something else. This could be done by having properties that hold information on the amount of legs the animal has, whether it is land-based, aquatic or aerial, and other factors. In this section, we go over commonly used methods for profiling or classifying network traffic.

2.3.1. Simple rule filtering

Within network traffic classification, there are many different ways to label the type of instances. The most-used application can be found in firewalls, where simple rule filtering is applied on inbound and outbound traffic to accept or deny packets in real time. They are preconfigured 'classifiers' that only look at packet-level values of traffic and are not able to analyse traffic patterns. Furthermore, they are often only placed at the edge of networks or near a DMZ (demilitarized zone), as those are the choke points for most malicious traffic sources.

If we wish to perform simple analysis on traffic for classification, we could use binning. The idea is that we select, change, create and group features that together give a clear distinction between expected and anomalous behaviour. Based on the resulting feature set , the value of that feature should then predict the class of the traffic instance. The set of feature values practically represents different categories an instance may belong to. To prevent too many categories from existing, every feature needs to be discretized to a manageable form. For all numerical features, it is sufficient to group the range of possible values in bins, for example 'small', 'medium' or 'large' for packet size.

Building such a model requires a strong analytical understanding of the data, as the feature set is manually built. Moreover, if the dataset has complex data, i.e. if the class instances seem to overlap in all dimensions, then this method might be unsuitable for obtaining good results. Lastly, it requires ground truths or labelled data to verify its effectiveness.

2.3.2. Machine Learning

In general, machine learning refers to the automatic learning or improving of algorithms through experience. A subset of machine learning focuses on making predictions using statistical data. The application of machine learning on (complex) problems is beneficial for several reasons. Foremost, the wide selection in machine learning algorithms allows for complex analyses to be executed to uncover patterns that are hidden from simple or manual analysis such as described in Section 2.3.1. Especially with high-level libraries, such as Keras[2], scikit-learn[3] and TensorFlow[4], machine learning has become more accessible and performing these analyses has become easier as well. Finally, with the exponential increase in available data and bandwidth, simple rule filters or manual analysis result in too many false positives (false alerts) or too much data too handle, making it infeasible for SOC analysts to keep up. In contrast, the scalability of some machine learning algorithms paired with the ability to perform complex analyses increases the effectiveness in terms of both time and accuracy.

Making predictions through machine learning can be done unsupervised or supervised. For unsupervised methods, there is no ground truth available for the data instances, therefore the algorithm must learn its own ideal feature space according to some objective function and by minimizing some loss function. Common unsupervised algorithms are clustering methods, e.g. k-means, and some applications of neural networks.

[2]`https://keras.io/`
[3]`https://scikit-learn.org/`
[4]`https://www.tensorflow.org/`

Supervised learning, on the other hand, uses labelled training data to guide its objective function. In the context of classification, the goal is to build a model that can identify the class of a new instance, based on its experience with the labelled training instances. For binary classification, there are only two classes for the problem. Multiclass classification is also possible and uses more than two classes. There are many types of algorithms, because there is no single form of classification that works for all types of data. The most commonly used are:

- Linear classifiers, which split the data in some dimensional space based on a linear combination of the used features;
- Kernel estimation, such as k-NN, which classifies new instances based on a probability density function on a random variable and is often combined with (other) probabilistic classifiers;
- Support vector machines (SVM), which can assign a class to new instances based on (non-)linear classification that automatically maximizes the distance between classes;
- Decision trees (and an ensemble of them called a random forest), which contain a root, branches and leaves, that lead each new instance to its predicted class based on the defined thresholds in each node;
- Some applications of neural networks, where the network is trained according to the actual label of a training instance.

Since we focus on interpretable results for this work, we will use decision trees and random forests for classification. The next subsections will explain how these algorithms work.

Decision Trees

A decision tree is a non-parametric supervised learning method used for both classification and regression problems. It reaches its goal by learning simple decision rules inferred from the training data (features). These decision rules split the data in segments based on feature values, such as splitting on the type of protocol used. A tree consists of a root node, where all data enters, followed by a finite amount of branches that split the data recursively based on decision rules to new nodes, eventually concluded by leaf nodes, where the data points are, in the context of classification, classified as the majority class of that leaf. Figure 2.3 shows an dummy version of a decision tree that attempts to separate benign from malicious traffic.

The non-parametric nature of a decision tree is defined in properties such as the maximum tree depth, minimum sample size for splitting, maximum amount of features to use and others. All these properties have significant effects on the classification score and potential overfitting of the resulting model. For each dataset, different parameter values may lead to better results. It is, therefore, important to select the optimal values for these parameters.

There have been multiple iterations of decision tree algorithms over the years. In 1986, Ross Quinlan developed the ID3[5] (Iterative Dichotomiser 3), which only allowed categorical features to be used. Its successor, C4.5, removed this restriction and improved upon the algorithm. The latest version is C5.0, but is under a proprietary license. For this work, we utilize the CART (Classification And Regression Trees) algorithm from the SKlearn Python library. This is a modified version of C4.5, which supports numerical target variables (regression). CART constructs binary trees and splits branches by using a feature and threshold value that maximize the objective function for each node.

The objective function is defined as the measured quality of a split. The higher the score on the objective function, the better a feature and its threshold are for determining the output class at that point in the tree. The two most common metrics for this function are the Gini impurity and information gain statistic.

Gini impurity measures the amount of impurity a decision rule brings to the subset of instances. More formally, it is a measure of how often a randomly selected element of a set is misclassified if randomly labelled according to the class distributions of a subset. A perfect impurity score is 0, indicating that the subset of elements in this split of the decision rule all correctly belong to one class. To compute

[5]`https://en.wikipedia.org/wiki/ID3_algorithm`

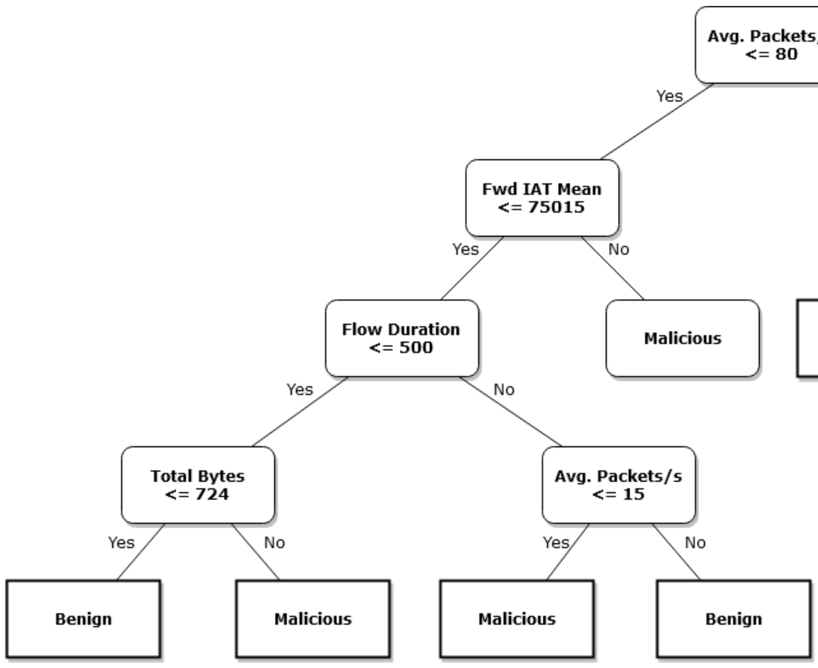


Figure 2.3: An example of a binary decision tree in the context of network traffic. The root node is at the top, with each decision leading it further down. Eventually, it hits a leaf node where it will be classified as the value inside the leaf node.

the Gini impurity for a set of elements with J classes where $i \in \{1, 2, ..., J\}$, let p_i be the ratio of items labelled with class i in the set:

$$I_G(p) = 1 - \sum_{i=1}^{J} {p_i}^2 \tag{2.1}$$

Information gain is based on the notion of entropy, and represents the amount of uncertainty or disorder that a split gives. It has a similar mathematical definition in the form of:

$$H(p) = -\sum_{i=1}^{J} p_i * \log_2(p_i) \tag{2.2}$$

where its main difference lies in the logarithmic part, which introduces a higher computational complexity. The maximum value is 1, representing full uncertainty, whereas the minimum value of 0, similarly to the Gini impurity metric, means the split provides a perfect subset with no uncertainty.

For the CART algorithm, a split depends on the feature variable type. For categorical features, a split is made for each value and the split with the highest objective function is taken. For numerical features, the rows are first filtered to only include unique values. They are then ordered by value, subsequently iterated over per pair and the mean of the two values is taken as the threshold value. Thus, $N - 1$ splits are attempted, where N represents the amount of rows, and the best split is chosen. This process continues recursively until no improvement can be made or until some parameter minimum threshold cannot be met.

Even with the aforementioned parameters to prevent overfitting, a decision tree is still relatively highly susceptible to have this issue. They can quickly scale out of control, splitting on seemingly arbitrary values to maximize the objective function on each level of the tree. One way to reduce overfitting is to limit the growth of the tree, e.g. by setting the maximum depth of the tree to a low value, or by increasing the amount of elements needed in a subset for a split to be valid, et cetera. Another option is to prune the tree post-training, where certain branches that do not meet the set conditions are removed from the tree. Both of these methods reduce the potential for overfitting, but generally do not give enough confidence to remove most suspicion of the model being trained for the dataset as opposed to the general problem. To combat this, a bagging solution has been designed called random forests.

Random Forest

A random forest is an ensemble of decision trees, tasked with fixing the instability and overfitting possibility of a decision tree. By providing each decision tree in a random forest with a different subset of the data and features, the overfitting of individual decision trees are cancelled by others. For each classification, the instance is passed through every decision tree and a majority vote decides the class label. Therefore, with enough decision trees in the ensemble, the chance of overfitting is greatly reduced. Moreover, instead of optimizing hyper parameters on a decision tree level, the goal is now to optimize parameters on a meta level to further enhance the diversity of the trees in the ensemble. Primarily, the amount of trees and the ratio of data given to each tree are the reasons that the potential for overfitting is heavily reduced.

2.4. Performance Metrics

The output of classification techniques are most easily represented in a confusion matrix, which summarizes every classified instance versus the actual class it belongs to in a x-by-x table, where x equals the unique amount of classes. An example can be seen in Table 2.1. In the domain of network traffic analysis, a positive instance is seen as a malicious instance, i.e. something that is worth analysing and taking action on, whereas negative instances are equivalent to expected, benign traffic. From a confusion matrix, most statistical evaluation metrics can then be trivially computed.

		Predicted class	
		Positive	Negative
Actual Class	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 2.1: Example of a confusion matrix.

The most reported metric in research is accuracy, which is the proportion of data that is correctly classified. When classes are balanced, this is a good measure. However, since anomalous behaviour in network traffic forms a very small percentage of total traffic, there is a very significant class imbalance and the usefulness of this metric deteriorates strongly.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

A less popular but still frequently used metric is the F-score, or F_1 score, which is defined as the harmonic mean of precision and recall (True Positive Rate). Due to the multiplicative property, it is sensitive to both low precision or recall. Often, it is used in cases where there is a large amount of actual negative instances and can therefore provide more information than the accuracy. In network traffic detection, anomalies are often portrayed as positive instances, meaning that the F1-score is a better metric in this context than accuracy, but it still does not give a complete picture.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.4}$$

$$\text{Recall/TPR} = \frac{TP}{TP + FN} \tag{2.5}$$

$$F_1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{2.6}$$

To combat this issue, other metrics such as Negative Predictive Value (NPV) or True Negative Rate (TNR) can be used, as the nominator contains the correctly classified negative instances. In particular, TNR, also known as specificity, measures the proportion of negative instances that are correctly classified, i.e. proportion of true negatives, whereas NPV measures the proportion of *predicted* negative instances that are correctly classified.

$$\text{Negative Predictive Value (NPV)} = \frac{TN}{TN + FN} \tag{2.7}$$

$$\text{Specificity/TNR} = \frac{TN}{TN + FP} \tag{2.8}$$

Another interesting indicator is balanced accuracy, which normalizes the true positive rate and true negative rate. The advantage of this is that it applies equal importance to the positive class and the negative class. However, it remains important to use multiple metrics, as it does not become clear from this metric where the classification scores are lacking.

$$\text{Balanced Accuracy} = \frac{TPR + TNR}{2} \tag{2.9}$$

Finally, a more advanced metric is the Matthews correlation coefficient (MCC), also known as a special instance of the phi coefficient. The coefficient takes into account all values in a (binary) confusion matrix and is a balanced metric which can be used even with imbalanced classes. Essentially, it is a correlation coefficient between the predicted and actual classifications. Its value ranges from -1, indicating total misclassification between predicted and actual instances, to +1, indicating perfect classification. It gives a balanced representation of the results, because it is the geometric mean of the regression coefficients of the classification problem and its dual.

$$\text{Matthews Correlation Coefficient} = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.10}$$

A quick example of MCC's usefulness can be shown with classification results suffering from data with class imbalance[6]. Consider the confusion matrix with the following values: TP = 90, FP = 4, TN = 1 and FN = 5. The accuracy and F_1 score, two of the most used metrics in machine learning, would be calculated at 91% and 95.25%, respectively, while the MCC value would only be 0.14, indicating that the model is far from perfect. Noteworthy is that the F1-score depends on which class is defined as the positive class, but this ambiguity allows misleading results to be reported. If we compare this to the balanced accuracy metric, which would give us a score of 0.57, we see that both metrics bring insufficient results to light regardless of class imbalance, but MCC is the geometric mean whereas balanced accuracy is the arithmetic mean.

Ultimately, the best metrics for a given problem heavily depend on the context. In the context of network traffic anomaly detection, a heavy class imbalance is often present in favour of benign traffic over malicious traffic. Therefore, careful consideration must be taken to select metrics that do not misrepresent the results. Moreover, The importance of Type I vs. Type II errors also affects which metrics should be used. As explained in Chapter 1, both errors are considered important, as false negatives indicate the IDS is unable to detect actual attacks, whereas false positives cause unneeded work and fatigue among SOC analysts. Thus, it is important to select metrics which can handle class imbalance and assign appropriate weight to type I and type II errors.

[6]Example taken from [4]

3 Dataset Design Analysis

Sommer and Paxson [35] stated that the most significant challenge an evaluation faces is the lack of appropriate public datasets for assessing anomaly detection systems. Often, new datasets are designed based on criticism of earlier datasets, where the authors use their own metrics on measuring how realistic their data is. The lack of consistent metrics and application thereof in the research field give way to suboptimal datasets, most of which are already considered dated and having unrealistic events after a year.

The goal of this chapter is, primarily, to find potential issues or inconsistencies in the creation of an IDS dataset and understand why they exist. Based on that, we could offer insight into how to solve these issues. To ensure a high enough confidence for the analysis, we considered the top 15 cyber security conferences[1] from 2016 up and till 2020 for relevant papers, as well as targeted search for dataset (overview) papers in the field of IDS. Interestingly, none of the conferences have papers that studied this issue.

We expect our approach to accurately answer our research question due to the vast amount of literature searched. Apart from considering top conferences as potential sources, we also performed an additional search for relevant papers outside conferences. In particular, we looked for studies which proposed new datasets or provided an overview of existing datasets. Moreover, for some relatively lowly-cited papers, we inspected the works that cited it for relevance.

In this chapter, we will answer Research Question 1:

Are the sets of quality metrics used by dataset creators suitable for creating realistic datasets?

To do this, we will present a chronology of related work on this topic and analyse the differences in chosen quality metrics or methodology. Although most of these works share basic similarities in the properties they deem to be important for datasets, such as the volume of data, relevance and variety of attacks, and others, it is startling how little consistency there is in the evaluation and creation of datasets. A dataset might be praised for its ingenuity in one, yet rejected in a later analysis. Beyond that, we will present our own observations from the analysis in a concluding section.

3.1. Similarities and changes over the years

Both the methodologies on the design of datasets and corresponding meta-papers, which present an overview and analysis of available datasets, have experienced vigorous changes over time. In this section, we will go over those changes and study the effectiveness of them. In particular, we first study the changes in proposed dataset quality metrics over time. Then, we will apply those metrics to available datasets based on the publication year. Finally, we will briefly reflect on the recent analysis by [14].

3.1.1. Quality Metrics

Our literature review starts with [34] in 2012. They worked on a systemic approach to creating new IDS datasets after noticing the scarcity of appropriate IDS datasets. Older datasets often used static

[1]as defined in `http://jianying.space/conference-ranking.html` retrieved on 27 January 2021.

data and did not allow for dynamic behaviour and events. These datasets are essentially a snapshot in time, are unmaintainable, inextensible and irreproducible. Often, by the time a dataset was released, it was already outdated against the current threat landscape. Their answer was to introduce dynamic profiles for generating predefined behaviour in traces. Each profile contains an abstract representation of wanted behaviour, characterizing a service or user type on the network. For example, a profile could be made for a host that browse the internet over HTTP during work hours. Existing data could be studied and abstracted to obtain the distribution of packets, the packet length, duration, and other factors. These factors could then be used to build the profile that will be used by programmed agents to generate traffic. The authors provide guidelines for creating an IDS dataset. Notably, post-capture data insertions are frowned upon, for example by artificially adding attacks from a different environment, as they are unnatural additions that do not necessarily fit the dataset. Moreover, they argue for complete information and attack variety. The push from static to dynamic datasets is an important step, as it combats the staleness of available datasets by generating a new one with updated distributions of user and attack behaviours. With this vision, they created a new dataset called UNB ISCX. Their approach opts for a realistic network architecture and traffic. Irrelevant properties in the data will only distract the analysis. Moreover, artificially added data will only negatively affect the quality of the work, according to the authors. The dataset should be labelled (correctly), as multiple detection methods depend on labeling. Completeness of information is also considered of vital importance, thus the authors will include all network interactions within or between the internal LANs. Privacy is of a lesser concern, due to the dataset being simulated from behaviour profiles on a controlled testbed architecture. Since sanitization of data on concerns of privacy removes usable information, the researchers decided to keep the data as is. Finally, diverse intrusion scenarios are required that capture the wide variety and intensity of the threat scope. The issue with ISCX's released dataset, apart from its age, is that the distribution of the user profiles is not based on real world statistics, meaning that the obtained classification results will not necessarily carry over to other datasets in the same domain.

In 2015, [20] presented their view on what a good IDS dataset should have. According to them, a good dataset should be *recent*, *labelled*, and *rich* in data. Rich in data implies a wide variety of threats, but also a sufficiently long capture time. For nearly all domains, a single trace of only a day or even less does not capture all possible benign traffic. Finally, it should be *correct*. Although very broad, the authors define it as the following: A dataset holds a benign component (background traffic) and a malicious component (anomalous behaviour). Any of these two components can be real (observed in a network) or synthetic (simulated according to a model). Furthermore, the components can be *correlated* (produced in the same experiment environment) or *unrelated* (combined from different environments). Ideally, a dataset holds real and correlated data that meets the aforementioned criteria. However, this is often an unrealistic goal for various reasons. Thus, synthetic or unrelated data should not be rejected, instead the validation of such data against 'real' data is the real challenge. Although the features mentioned here are somewhat broad and unquantified, it does not seem to explicitly require an analysis of the threat domain, i.e. what attacks are relevant in the infrastructure that is supposed to be simulated/emulated. Some attacks happen more frequently or more intense in certain domains over others. Moreover, there are attacks that are specifically crafted for certain types of infrastructure and do not show their true behaviour until exposed to such an environment.

[11] presented a new framework in 2016 on evaluating the quality of datasets. It consists of 11 metrics revolving around the experiment setup, completeness of information and reported output. In detail, the authors believe that a complete network configuration is necessary for some attacks to show their true behaviour. Moreover, the data should be complete in all aspects, i.e. no filtering of seemingly useless traffic or interactions, have a relevant set of available protocols used, and enough attack diversity. Labelling is considered important as well, as is minimizing anonymity to allow for payload-inspecting detection mechanisms. The heterogeneity of output data can best be described as multiple different output types. Especially in the domain of generating network traffic data, collecting host logs or network equipment logs could result in more ways to detect anomalous behaviour; at the very least bringing more context to each anomalous instance. Finally, a rich feature set and metadata documentation should be provided, as the main goal of the dataset is for other researchers to test and analyse their proposed methods. Thus, having more information available will allow researchers to perform their work in a more effective and efficient manner. To evaluate the datasets, they provide coefficients for each metric, with higher coefficients for metrics such as the amount of protocols, attacks and completeness of traffic. Each metric coefficient is then evaluated on a binary basis and summed to

provide a final score between 0 and 1. Most of the author's 11 reported quality metrics are discussed in previous papers. The call for proper documentation and heterogeneous output data is less used as metric, although still useful. A surprising find is the lack of age relevance. Their evaluation framework is tested on old datasets, such as KDD99 and KYOTO, receiving respective scores of 0.56 and 0.85. The authors do not expand on the interpretation of the scoring value range, but it is fair to assume that the latter score carries a positive meaning. Since anonymity and heterogeneity of data are active issues in network traffic anomaly detection, nearly all datasets will score 0.1 less on the evaluation. Most detection methods can be done on network flow traffic alone, however, removing the need for solutions to those issues.

[12] proposed a dataset quality evaluation metric using a fuzzy logic system based on the Sugeno fuzzy inference model [37]. The metric is based on two input sets X and Y, where X represents the set of six equally important properties of datasets based on related works, such as the maximum number of possible attacks included, current attack behaviours, availability of ground truth information, real-world normal traffic dynamics with relevant timings and complexity, and more. However, these properties do not seem to be well-defined. One property judges datasets by the maximum number of possible attacks, but it does not become clear what the maximum is in this context. The same goes for current attack behaviours and maintenance of cyber infrastructure during capture. Completeness of data, real-world traffic dynamics and labelling, the three other properties, are found in most other works and are clear in their definitions. The set Y only contains two elements, representing the generation environment of the data. A production / real network provides a higher score than a synthetic network or testbed, which is understandable, as synthetic data is much more difficult to validate as representative than to real captured data. The final output of the model, R, is the weighted average of all rule outputs, and represents the probability of the dataset being realistic. This metric will benefit from more detailed properties. For instance, current attack behaviours could be renamed to 'representative events'. An event is representative if it fits the current domain space of the dataset. The maximum number of possible attacks property could then focus on the frequency and variety of malicious events, given the current domain space. The metric was tested on six datasets with statistical analysis results from other papers, including DARPA [21], ADFA-LD [6], Kyoto [36], ISCX, DEFCON, LBNL and CAIDA [34]. None of the datasets had a good score, with ISCX having the highest at $R = 0.47$. With this in mind, the authors set out to create a new dataset, NGIDS-DS, which focuses on maximizing the metric score. The new dataset obtained an R-value of 0.95 with a focus on host-based intrusion detection due to the labelling technique.

[28] contributed a literature survey of datasets for publicly available network-based intrusion detection datasets released in 2017 or earlier. They describe each dataset in detail and identify 15 different properties to assess the suitability of individual data sets for specific evaluation scenarios. Their evaluation is not based on a score - unlike other works[11, 12, 31]. Their reasoning is that the importance of certain properties depend on the context and domain of a dataset. The properties are summarised in five categories: General information, nature of data, data volume, recording environment, and evaluation. According to the authors, the most important properties were the labelling and the heterogeneity (packet-/connection/host-level) of datasets, which are in the evaluation and nature of data category, respectively.

3.1.2. Dataset design methodologies

AWID [17] was created for wireless network anomaly detection. The authors focus their efforts on providing a holistic variety of known wireless attacks (WEP) in a controlled lab setting. They provide a full and reduced setting, both having binary and multiclass labelled versions. The reduced version had a capture time of one hour, of which 35 minutes were devoted to malicious activities. It seems to fit most of the criteria listed above, yet it lacks in the methodology. For one, the environment is build similar to a single lab room with multiple wireless devices, which is meant to replicate a small (home) office environment. Secondly, the behaviour of the devices, i.e. the benign background traffic, was not described in detail, and therefore may be lacking as it might not represent the full range of wireless background traffic. Finally, there was only one attacking device and it focused solely on WEP attacks. Although its attack focus is unusual, it is not necessarily unsuitable, as there aren't many similar datasets for this specific focus.

UNSW-NB15's infrastructure attempts to emulate an enterprise environment, albeit on a much smaller scale. The designed attacks were chosen specifically for this environment, and two traces were captured each with a different attack frequency. Great effort is put into labelling and describing the data - as seen further in this work in Figure 4.3. However, the pcap files that were generated were transformed into network flows and thrown away, potentially destroying valuable information that could be used for packet-level anomaly detection. Despite of this, the dataset was good at its time, but is now deprecated due to the changes to the threat landscape.

Also in 2017, [26] created CIDDS-001 and CIDDS-002. Their metrics were heavily based on [20], but include an explicit guideline to consider network topologies, i.e. what domain is supposed to be mimicked. In both cases, the domain is a small business environment, where CIDDS-001 also has an external server open to the internet. Their methodology does not use the broad definition of proper variety in attacks, but instead opt for problem specific data, where the creation of a dataset is tailored to the algorithm to be tested. CIDDS-002 is a prime example of this, as it only contains port scans as malicious data. CIDDS-001, on the other hand, contains DoS, brute force and port scans as the set of attacks. In both cases, the design metric of problem specific data makes the datasets very niche and unrepresentative for most network configurations, but nonetheless may be useful if the threat landscape only contains those attacks.

The UNB IDS datasets from 2017 and 2018 both use the notion of B-profiles as introduced by [34]. The design metrics and methodology were based on the framework of [11] and the authors follow them very well. [32] provides an extensive overview of publicly available datasets until 2013. They analyse each dataset for its usefulness and conclude that all datasets suffer from volume, distribution or variety of attacks issues. Moreover, they contribute a new dataset that focuses on these issues and eleven other identified criteria introduced in an earlier paper [31]. For instance, their framework requires datasets to be generated using a complete network configuration, consist of complete traffic, be labelled, have attack diversity, and more. [39] critiqued the datasets briefly on a few points. Their first point was about the large amount of instances, as processing the dataset files is a tedious task compared to smaller datasets. This is controversial, as other studies so far have promoted the completeness of information and, implicitly, the longer runtime of the experiment. Moreover, they preferred each file to contain all attack labels for processing, but understand that combining the files leads to more computing and processing time needed. One justifiable critique is about the missing and redundant data records. A reasonable amount of instances are missing data values for certain features, requiring future researchers to deal with that before being able to test their methods on the dataset. In most cases, this would be dealt with by simply throwing the incomplete instances away, but this goes against the completeness of data metric, and it might affect the sequential property of network traffic. Finally, they argue against the high class imbalance that the dataset has, fearing it might result in unrepresentative classification scores. We disagree with this point, since the problem statement of network traffic anomaly detection has inherent class imbalance. If we strive towards creating realistic datasets, we require realistic traffic distributions as well, which include the proportion of benign vs. anomalous traffic. Instead, researchers who use these datasets should be aware of the class imbalance and report more meaningful metrics, as discussed in Section 2.4.

Advancements in network flexibility gave rise to Software Defined Networks (SDN). These represent a new domain where greater control exists over the network. The main benefit is the separation of the control plane from the data plane, i.e. separating decision making from forwarding traffic, which is still found in traditional distributed neworks. It also allows for a central view of the entire network through an SDN controller and provides easier access to network (infrastructure) upgrades or changes. [9] reviewed and classified possible attacks in different SDN layers. Moreover, they studied the limitations of existing IDS datasets and propose a new testbed to generate datasets for the SDN domain. Finally, they generate a dataset with varying attacks and demonstrate how to apply machine learning algorithms on it for anomaly detection. Their study on other datasets does not provide detailed information. The biggest criticisms were about the dataset age, event relevance and variety in attacks. For their own dataset, they use seven different attack types.

3.1.3. Current state-of-the-art quality metrics

Kenyon et al. [14] did an analysis in 2020 on the quality of publicly available IDS datasets. They frequently identified key issues with intrusion detection datasets, including poor provenance (unclear methodology), unclear data composition (benign/malicious event frequencies and scope), poor data quality (duplicate data, unrepresentative samples), over-summarisation (potential loss of information), unmaintained datasets (threat event signatures become unrepresentative after some time), domain specificity (unrepresentative domain for commercial deployments), and under-represented domains (e.g. IoT/ICS networks). They included nearly all publicly available datasets known for (network) intrusion detection. For each dataset, they summarized its features, provided an analysis based on the key issues they identified and other properties, and presented a final verdict on the usability of each dataset. Their analysis reveals ten significant flaws, translated to the feature set in Figure 1.2, that most methodologies suffer from. In that feature set, they recommend data provenance as the most important feature when designing datasets. The other features, whether considered mandatory or desired, coincide with previous papers on the topic. The authors conclude that many widely used datasets are either out of date, or not representative of today's threat landscape in the respective domains. Therefore, most datasets are unusable, and the value of using them in (academic) work is questionable.

3.2. Conclusions

In this chapter, we have performed a literature review of papers about dataset quality metrics since 2012. It becomes clear that this field still lacks consistent design metrics and methodology, even though it has been active for over two decades. From this overview, we can draw several direct and indirect conclusions with respect to the research question and sub questions:

- There is a general lack of knowledge and agreement among researchers regarding best practice metrics and design choices when creating a dataset.

Researchers that create a new IDS dataset very rarely analyse the current state-of-the-art in dataset design methodology, primarily because there is no agreed upon state-of-the-art work available. They base their own design metrics on outdated information of older datasets and criticisms thereof. The result is a dataset that is marginally better than what is currently out there, but still lacks on multiple aspects. To partially solve this issue, we provide a summarised overview of the studied papers from Section 3.1.1 in Table 3.1. We strongly suggest dataset creators to consult those papers for more in-depth information, prioritizing newer works over older ones. If newer datasets adhere to the extensive collection of quality metrics, then we can analyse the usefulness of those metrics and get closer to a methodology that publishes representative datasets.

- Understanding the target domain is vital in creating a representative dataset.

Both the network infrastructure, traffic and threat landscape vary significantly between domains/industries, yet this property is often under-analysed and insufficiently described in papers. Dataset creators often opt for network infrastructures that are easy to create, e.g. SOHO (Small Office / Home Office) environments, yet use the threat landscape of different infrastructures, such as multi-department businesses.

- Lack of data provenance severely hinders future work on a dataset.

Classifying only a single dataset disallows researchers from drawing conclusions that can be applied to the bigger picture, for instance the performance on similar datasets within the same domain and threat landscape. Most researchers, therefore, test their algorithm on multiple datasets to assess its performance. Due to poor data provenance and an arbitrary selection of datasets, the conclusions still cannot be applied to a broader view, but instead is limited to only the datasets tested.

With these conclusions, we identified the largest issues with creating and evaluating IDS datasets. What quality metrics are relevant, which are more important than others and how to meet the metric requirements are important questions that do not have a clear answer. To aid with this, we provide a list of five important properties that are often overlooked or insufficiently met:

1. **Domain Context**: Traffic composition varies wildly between domains/industries, yet a thorough analysis or definition of the domain is often lacking. What domain is specifically being targeted on which scale? What event distributions are used (e.g. how many machine sensors reporting data in a SCALA setting)? We urge researchers to more explicitly analyse and define their domain, such that it becomes clear what type of industry it covers and how it does so;

2. **Data Provenance**: The methodology and properties of a dataset is generally insufficiently described. Besides the points mentioned by Kenyon et al. [14], additional attention should be paid to properly describe the infrastructure, as it is not always clear what it simulates and how the devices interact with each other;

3. **Relevant threat landscape**: Extension of property one. Threat landscapes vary wildly per context, so including very general or outdated attacks such as HTTP DoS attacks will not result in a representative threat list for most contexts. One possibility for improving the threat landscape is to consult recently reported hacking events within the same domain. These reports illustrate which relevant attacks are active within the industry and give insights to the prevalence of specific attack vectors;

4. **Relevant benign behaviour**: Also an extension of property one. Most organizations use streaming software for remote meetings, yet datasets commonly only use e-mailing and web browsing as benign activities. Trivial benign traffic will break each IDS if updated with new benign behaviour, such as new workstations with a different operating system. We recommend looking into additional benign activities and used operating systems for those activities, as both provide different traffic patterns;

5. **Update support**: Even if the dataset is fully representative, it still remains a snapshot in time and will be outdated a few years down the line. Providing a framework that can generate updated versions would, therefore, retain much more value for researchers. As a baseline, the work by Shiravi et al.[34] introduces the notion of agent profiles, dedicated to generating specific traffic types. These could be introduced in the dataset creation methodology to allow more flexibility in altering agent behaviour, instead of creating static behaviours that are unable to adapt or added upon.

Additional attention should be paid to the above properties when creating a new (network) intrusion detection dataset or judging the quality of an existing dataset. They also answer our research question, whether the current quality metrics are suitable for creating a realistic dataset. Based on the findings, we conclude that, although there have been improvements over the years in chosen quality metrics, the set of metrics is still insufficient for creating a realistic dataset. By focusing more on the metrics listed above, we believe better datasets can be designed for future state-of-the-art research.

Paper	**Year**	Description
Kenyon et al. [14]	2020	14 features, of which nine are considered mandatory. Clear methodology and domain description are considered as the most important features. Authors analysed over 20 publicly available IDS datasets to draw these conclusions.
Ring et al. [28]	2019	15 features grouped in five categories: General information, nature of data, data volume, recording environment, and evaluation. Proper labeling and heterogeneity of information were the most important features, according to the authors. Their data includes datasets released in or before 2017.
Haider et al. [12]	2017	Quantified scoring scheme using two input sets X and Y. X contains six equally important metrics for evaluating datasets, and Y describes the origin of the data (real capture vs. synthetic generation).
Gharib et al. [11]	2016	Quantified scoring scheme using 11 features, each with their own scoring weight. Due to the broadness of information completeness and attack diversity, these have been assigned a greater weight.
Malowidzki et al. [20]	2015	Four general properties that define a good dataset. Authors provide a brief explanation for the first three properties: *Recent*, *labeled*, and *rich*. For the last property, *complete*, a larger description is given.
Shiravi et al. [34]	2012	Introduced the notion of using profiles to generate traffic for synthetic datasets based on behaviour types. Argues for dataset creators to focus on a framework to create datasets, instead of creating a static, unmodifiable dataset that essentially is a snapshot in time.

Table 3.1: Literature study overview of dataset design/evaluation metrics.

4 Dataset Classification Complexity

In this chapter we will explain our process to answer Research Question 2:

> Is there a relationship between the design quality of a dataset and the difficulty of obtaining state-of-the-art classification results?

We do so primarily by focussing on its first subquestion:

> Is it possible to create a general white-box classification model that works well on multiple network intrusion datasets?

Section 4.1 describes the chosen datasets for this experiment, along with state-of-the-art prior work that classify on these datasets. After, we provide an overview of related work that help with feature selection in Section 4.2. Then, Section 4.3 presents the methodology behind the classification pipeline, from data preprocessing to feature selection, model tuning and metrics that will be reported. Finally, we present the results for classification on each dataset in Section 4.4 and answer the subquestion.

With the results, we can also answer the second subquestion:

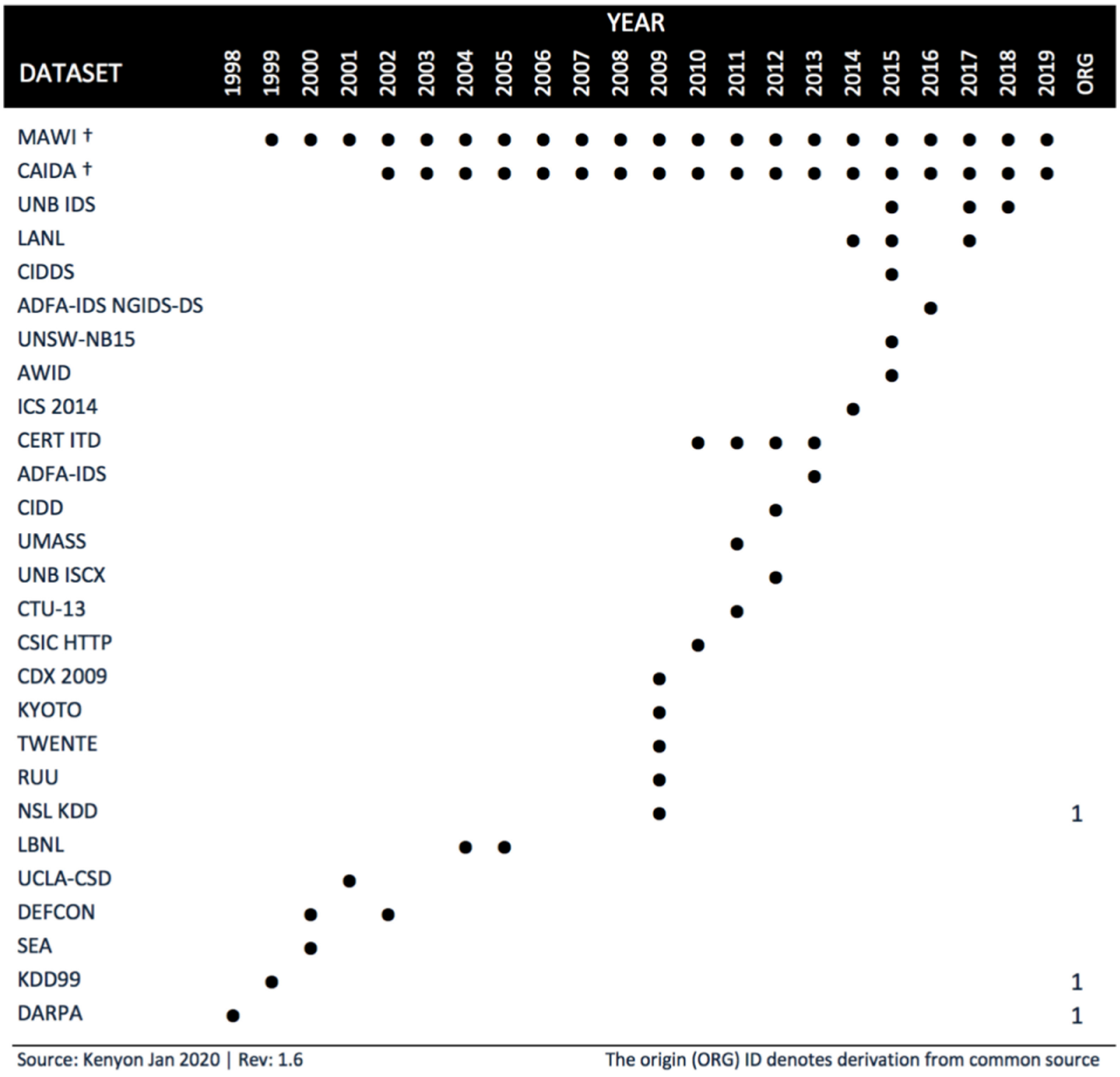
> Are state-of-the-art solutions by other researchers overcomplicated, i.e. are our results similar or possibly even better than theirs?

By comparing the classification scores of prior work with our white-box model, we can draw conclusions on the differences between the scores and methodologies.

4.1. Data

Before we are able to construct a general pipeline for our algorithms, we need to inspect the datasets. Since we will not optimise the algorithms for each dataset, it is important that we understand the nuances of each dataset, the differences and the similarities. Any findings will aid significantly in the process of creating our pipeline, such as the preprocessing techniques to use, the general patterns found in the data, and the available features. This chapter will provide an overview of the chosen datasets and two corresponding state-of-the-art solutions utilizing black-box machine learning techniques - if available, along with an analysis on the used techniques and data.

To broaden the data scope and obtain more conclusive findings, it is preferred that our classification pipeline handles both levels of traffic analysis: packet-level and connection-level. It is, therefore, beneficial if most datasets can be preprocessed to work on both levels of analysis. Moreover, the datasets should be labelled, as we will only explore supervised techniques in this work. We further filtered the publicly available datasets on their age, meaning that we only consider intrusion detection datasets produced in 2015 or later for this work. Our argument for this is based on the dataset provenance and event relevance quality features as described in Figure 1.2. In particular, we believe that older datasets contain event patterns that are considered too simple or are irrelevant by now; a technique that works well on those older datasets will often not necessarily work well on newer datasets. Moreover, we will only consider datasets that focus on network data and are synthetic (simulated in a testbed environment). Some intrusion detection datasets focus on real/host data instead, but this broadens the research scope too much. We disregard datasets which feature sets are too different from the other datasets, as that makes it difficult to generalize well. Finally, we require the set of datasets to have a broad domain context and event diversity, so our pipeline will generalize properly.



DATASET	YEAR																						
	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	ORG
MAWI †		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
CAIDA †					●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
UNB IDS																		●		●	●		
LANL																	●	●		●			
CIDDS																		●					
ADFA-IDS NGIDS-DS																			●				
UNSW-NB15																		●					
AWID																		●					
ICS 2014																	●						
CERT ITD													●	●	●	●							
ADFA-IDS																●							
CIDD															●								
UMASS														●									
UNB ISCX															●								
CTU-13														●									
CSIC HTTP													●										
CDX 2009												●											
KYOTO												●											
TWENTE												●											
RUU												●											
NSL KDD												●											1
LBNL							●	●															
UCLA-CSD				●																			
DEFCON			●		●																		
SEA			●																				
KDD99		●																					1
DARPA	●																						1

Source: Kenyon Jan 2020 | Rev: 1.6 The origin (ORG) ID denotes derivation from common source

Figure 4.1: Overview of public datasets analysed by [14]. For each dataset, the year of release and potential origin is given. Note: † marks long-term archives of Internet backbone data.

Kenyon et al. provide a recent overview of publicly available intrusion detection datasets, as seen in Figure 4.1. With the above requirements and overview in mind, the following publicly available intrusion detection datasets were chosen:

- **UNSW-NB15**, created in 2015 by researchers from UNSW Canberra Cyber [23], was generated with a traffic generation tool called IXIA PerfectStorm [1]. It contains four scenarios and nine types of attacks.
- **CIDDS** [26, 27] simulated a small business environment in 2017. For the malicious traffic, they included outside attackers who performed three types of attacks and allowed access to an external

[1]`https://www.keysight.com/zz/en/products/network-test/network-test-hardware/perfectstorm.html` Retrieved on December 17, 2020

server through the public internet, exposing it to real and up-to-date attacks.

- **UNB IDS** has a collection of IDS datasets for several domains and data types. Datasets concerning the Android domain, DNS over HTTPS (DoH) traffic, simulated corporate networks on AWS and others are included. For us, CICIDS 2017/2018[2] [32] and CIC-DDoS2019 [33] are of interest. According to [14], the quality of the CICIDS 2017/2018 datasets, compared to the other datasets listed in Figure 4.1 is above-average.
- **inSDN** [9] is a unique dataset released in 2020 designed for Software-Defined Networks. An extensive literature review has been done on similar SDN datasets and the researchers propose solutions in this dataset to the given criticisms on earlier work.

A more in-depth analysis along with corresponding state-of-the-art solutions will be presented in the subsequent subsections. For complete information about each dataset, it is recommended to read the original papers cited in the text. A quick overview of the datasets can be seen in Table 4.1.

Dataset	Year	Origin	Included threat types											
			REC	AUT	DoS	BOT	SPM	WRM	WEB	BCK	SHEL	SSH	DNS	OTH
InSDN	2020	University College Dublin	■	■	■	■	-	-	■	■	-	-	-	■
CIC-DDoS2019	2019	University of New Brunswick	-	-	■	-	-	-	-	-	-	-	-	-
CSE-CIC-IDS2018	2018	University of New Brunswick CSC & CIC	■	■	■	■	■	-	■	■	-	■	-	■
CICIDS2017	2017	University of New Brunswick	■	■	■	-	■	-	■	-	■	■	■	■
CIDDS-001	2017	Coburg University, Germany	■	■	■	-	-	-	■	-	-	-	■	-
UNSW-NB15	2015	University of New South Wales	■	-	■	-	-	■	-	■	■	-	■	-

Table 4.1: Overview of chosen datasets and included threat types.

Dataset	Year	Benign (%)	Malicious (%)	Available data levels	Size (GB)
inSDN	2020	20	80	Netflow	< 1
CIC-DDoS2019	2019	99.89	0.11	Packet, Netflow	31
CSE-CIC-IDS2018	2018	83.07	16.83	Packet*, Netflow	6
CICIDS2017	2017	80.3	19.7	Packet**, Netflow	1
CIDDS-001	2017	91.6	8.4	Netflow	4
UNSW-NB15	2015	87.35	12.65	Packet**,Netflow	< 1

Table 4.2: Dataset composition overview. Size per dataset is measured on netflow data. *Packet data cannot be labelled. **Packet data could not be downloaded successfully.

4.1.1. UNSW-NB15

Figure 4.2 provides an overview of the UNSW-NB15 testbed used to generate the traffic. The infrastructure contains three virtual servers, one of which infected with malware and thus generating malicious traffic. The set of attacks include DoS, fuzzing, worms, shellcode, backdoors, exploits, reconnaissance, analysis and some other generic attacks. Each server is connected to a router, which itself serves multiple other clients.

The connection-level dataset consists of two simulations, one of 16 hours and one of 15 hours. The traffic generation tool, IXIA, is configured to generate one attack per second for the first simulation and ten attacks per second for the second simulation. Figure 4.3 shows the data composition per simulation. From this, the distinction in attack frequency becomes much clearer, while the overall amount of data stays relatively the same. The authors configured a partition of these two simulations as a train/test

[2]`https://registry.opendata.aws/cse-cic-ids2018/` Retrieved on 18 December 2020

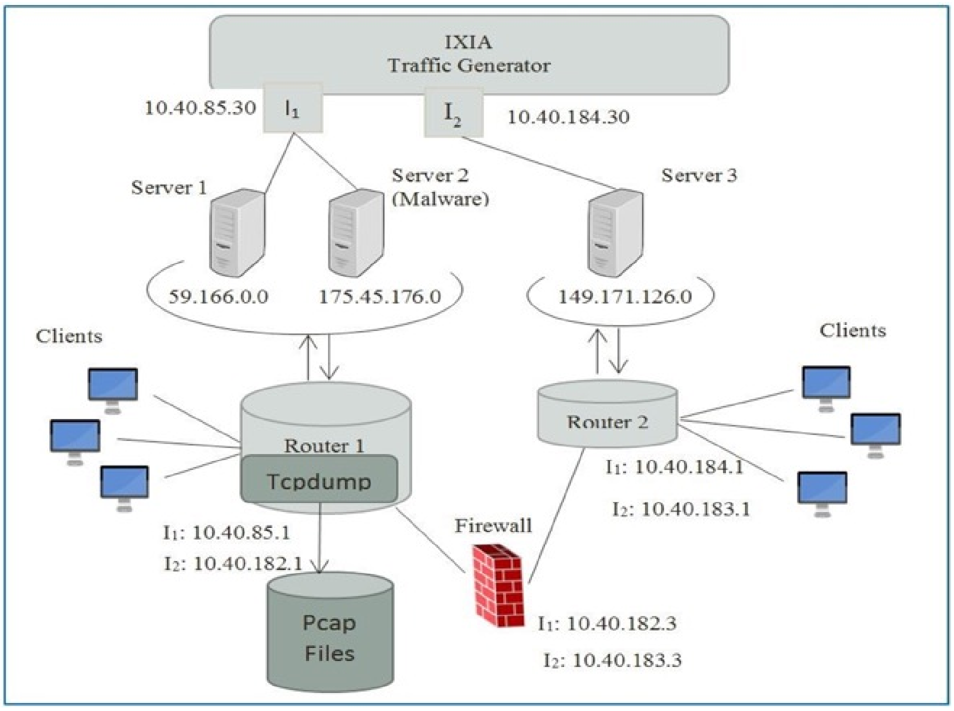
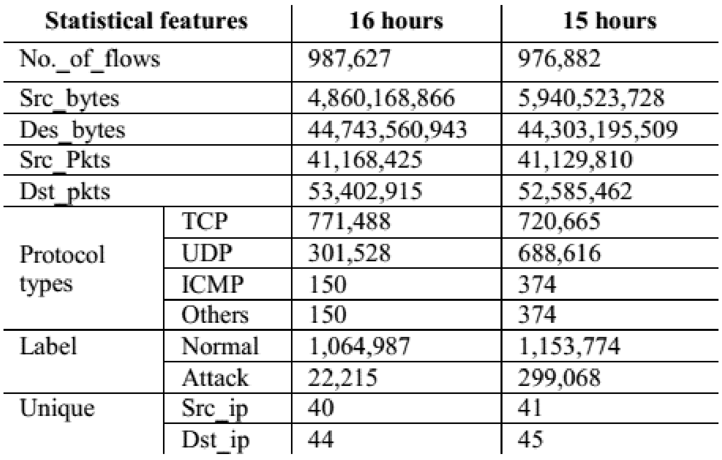


Figure 4.2: UNSW-NB15 testing infrastructure.

split. Since relevant papers use this partition in their classification models, we use it as well to ensure a fair comparison.

The original pcap files are also available for download. However, the downloads for some of the files repeatedly failed after multiple attempts, thus we cannot use them for packet-level analysis.



Statistical features		16 hours	15 hours
No._of_flows		987,627	976,882
Src_bytes		4,860,168,866	5,940,523,728
Des_bytes		44,743,560,943	44,303,195,509
Src_Pkts		41,168,425	41,129,810
Dst_pkts		53,402,915	52,585,462
Protocol types	TCP	771,488	720,665
	UDP	301,528	688,616
	ICMP	150	374
	Others	150	374
Label	Normal	1,064,987	1,153,774
	Attack	22,215	299,068
Unique	Src_ip	40	41
	Dst_ip	44	45

Figure 4.3: UNSW-NB15 data features.

Faker et al. [10] constructed three machine learning models for the NB15 and CICIDS2017 datasets. A feed-forward deep neural network with 3 hidden layers and 41 features (DNN), a random forest (RF) implementation with 36 features and a gradient-boosted tree (GBT) with 26 features were used to train and test on the given datasets. Their feature selection process was described, but it is unknown which features were ultimately used. A 5-fold cross-validation split was used where 20% of each iteration was used for testing and the remaining 80% for training. For binary class classification (benign vs. malicious), DNN outperformed RF and GBT with an accuracy of 99.19% against 98.86% and 97.92%, respectively. For multiclass classification, all non-malicious flows were first removed such that the models only had to differentiate between different types of attacks instead of malicious vs. benign as

	Internal server (OpenStack)				External server			
	PortScan	PingScan	DoS	BruteForce	PortScan	PingScan	DoS	BruteForce
week1	16	10	11	5	0	0	0	0
week2	8	6	7	7	2	0	0	4
week3	0	0	0	0	5	0	0	7
week4	0	0	0	0	1	0	0	3

Table 4.3: Malicious attack count per week and server origin within the CIDDS-001 dataset. Each column describes a specific attack.

well. DNN again outperformed RF with an accuracy of 97.04% against 91.77%. Other quality metrics were not reported. Given the class imbalance present in NB15, it is therefore not sure how well the algorithms perform on all levels.

Khan et al. [15] proposed a two-stage deep learning model based on a stacked auto-encoder with a soft-max classifier. The model was evaluated on KDD99 and NB15. In both cases, the model only used 10 features, significantly less than [10]. The results are presented in a confusion matrix for binary classification. They report an accuracy of 89.13% and an F1-score of 0.9085.

4.1.2. CIDDS

CIDDS is a collection of two flow-based datasets, CIDDS-001 and CIDDS-002. In both cases, a small business environment was emulated using OpenStack. The business environment contains a developer subnet, an office subnet and an internal servers subnet, which are connected to the internet with a firewall in between. For CIDDS-001, an additional management subnet and external server (directly accessible from the internet) were included. CIDDS-001 contains port scans, ping scans, DoS- and BruteForce attacks, while CIDDS-002 only contains port scans, albeit more specific in their type. Due to the larger variety in traffic and attacks, CIDDS-001 is more often used in research. Thus, we will continue with CIDDS-001.

The data is captured over 4 weeks, split per week and server location (internal vs. external). Table 4.3 illustrates the attack distribution per trace. For the internal server, week 1 and 2 contain the attacks, where the other weeks have only benign data. The external server has only benign traffic in week 1, where the other weeks have port scan and brute force attacks.

Abdulhammed et al. [1] explored five machine learning models on this dataset:

- DNNs of three layers with three different hidden layer sizes of 512, 1024 and 2048 neurons, and a learning rate of 0.1;
- RF;
- Voting techniques on OneR, Naïve Bayes, and ExtraTree;
- Stacking technique with Linear Discriminant Analysis, Naïve Bayes, and OneR;
- Variational Auto-encoder.

They report a variety of metrics, including accuracy, false alarm rate, detection rate (DR), precision, recall and others. Specificity is not reported, however. DNN along with down-sampling and class balancing leads to effective results in terms of accuracy, with 99.30% and 98.77%, respectively. The random forest implementation seems to outperform the DNN on most distributions, attaining over 99.9% accuracy and DR, and a FAR of $1E-4$. The authors are not clear on the used features per algorithm, instead only providing a complete feature list of the dataset, which includes IP addresses and source/destination ports.

Another group of researchers proposed a cloud-based IDS using the Google Cloud Platform [13]. The first step involves splitting the data using a 60/40 train/test split and a time-based sliding window of 5 minutes. Each set is passed to the anomaly detection module using a Naive Bayes classifier. Finally, an ensemble learning classifier based on Random Forest is used to perform multi-class classification for the type of attack. They only used week 1 for training and testing as a 60/40 split. When compared to a standard Random Forest classifier, the results are satisfactory. An average accuracy of 97% over all attacks was obtained, together with a FPR of 0.21% and an average runtime of 6.23 seconds.

4.1.3. UNB IDS

CICIDS2017, its updated 2018 version and DDoS2019 contain both labelled flow traces and the original pcap files. Traffic flows are generated using CICFlowMeter[3]. For CICIDS2017, the downloads for some pcap files failed on numerous occasions, similar to the UNSW-NB15 pcap files. Moreover, for CSE-CIC-IDS2018, the flow based dataset does not contain enough information to link the packets to their respective flows. Therefore, we cannot use the pcap files in our analysis for these two datasets.

CICIDS2017

This dataset is an improvement over the original IDS 2012 dataset - referred to as 'ISCX', as ISCX included only XML flow summaries and the events were not representative of today's event distribution. Furthermore, the authors analysed real traces to build 25 synthetically generated user profiles. It contains 8 attack types used throughout the dataset and is recorded over a full work week, i.e. 5 days. The download has two versions, one with all generated flow columns by the CICFlowMeter and a separate reduced set. The latter does not have certain column, such as the protocol or IP addresses, so we disregard it and work with the full set.

Faker et al. [10] also trained and tested their DNN model on CICIDS2017. Again, a 5-fold cross-validation split was used where 20% of each iteration was used for testing and the remaining 80% for training. Moreover, they tested their models on two versions of the dataset, one with replacement of missing values and infinite values by default values, and one where those rows are removed entirely. Finally, for multiclass classification, they remove all non-malicious packets such that only the attack packets are kept. For binary classification, they reported an accuracy of 97.73%. For multiclass classification, their implementation achieved an accuracy of 99.56%. An implementation of Gradient-Boosted Tree (GBT) was also built and tested here. It performed the best with a binary classification accuracy of 99.99% on 21 features, but had no multiclass results due to technical limitations within their working environment. No other metrics were reported.

Ullah and Mahmoud [41] proposed a two-level flow-based hybrid model. The level-1 model uses a decision tree with empirically-selected flow-based features to classify traffic as benign or malicious. If the traffic is classified as malicious, the flow is forwarded to the level-2 model which uses Recursive Feature Elimination to select the best features, and Synthetic Minority Over-Sampling Technique and Edited Nearest Neighbours to oversample minority classes. They generate a 60/40 train/test split, although it is unknown how the split is exactly done. A random forest classifier is then trained using the edited training data and classified using original testing data. Interestingly, binary classification is measured with a specificity, precision, recall and F1-score of 100%, while cross-validation only changes the specificity to 99%. It should be noted that the authors are using source/destination ports in their feature sets. Ports often are a great indicator within network datasets, because of the limited amount of data. In a real setting, dependent on domain and threat landscape - of course, there would be enough diverse data that ports become mostly irrelevant for netflow anomaly detection. Looking at multiclass classification results, all benign traffic, (D)DoS attacks and most other automated attacks were detected with 99-100% accuracy. Brute force attacks were detected with an F1-score of 87%, and SQL injections, XSS attacks and infiltrations were even more difficult to detect, with often lower recall scores of 70% or less.

CSE-CIC-IDS2018

The upscaled and further-improved version of the 2017 version includes over 420 machines and 30 servers, logically separated into 5 departments with different user profiles. The attack infrastructure consists of 50 machines and 7 threat scenarios, and generated traffic over 10 days. AWS was used to facilitate this experiment and to store the logs. Aside from network traces, it also includes system logs for each machine to correlate events with. Table 4.4 shows the class distribution of the dataset.

In [18], an auto-encoder based on random forest feature selection was used to perform multiclass classification on the CSE-CIC-IDS2018 dataset. They used 85% of the benign samples to divide them into a sparse and dense matrix. The remaining 15% benign samples and 100% of the malicious samples were used for testing. To select the best features to use, they performed random forest feature selection for both the dense and sparse matrix. Both matrices showed different features to be important. Their raw data approach did well for some attack types, with (nearly) everything being classified correctly. However, for detecting bot behaviour, brute force attacks through web or XSS, SQL injection,

[3]`https://www.unb.ca/cic/research/applications.html#CICFlowMeter` Retrieved on 2 January 2021

Traffic Type	Distribution (%)
Benign	83.070
DDoS	7.786
DoS	4.031
Brute force	2.347
Botnet	1.763
Infiltration	0.997
Web attack	0.006

Table 4.4: Data distribution of CSE-CIC-IDS 2018.

infiltration and DoS attacks with the HULK DoS tool, about 30-50% of the classifications were incorrect. The unweighted mean accuracy over all classes corresponding to their findings on raw data is 81.92%. For their dense and sparse matrix, on the other hand, did significantly better for bot behaviour, XSS brute forcing and DoS attacks. An average unweighted recall of 83.82% for the dense matrix on available classes and 80.47% for the sparse matrix outperform the raw data recall rate of 58.38%. Their metric reporting style does not allow us to compare it to binary classification results, but it does provide valuable insight in attack-specific classification.

[16] developed multiple different IDS solutions with SVM, K-NN and DT algorithms. They sampled only a small percentage of data from each dataset, including CSE-CIC-IDS2018, perform min-max normalization on the features and then split the data in 10 separate folds for cross validation. They did not perform any feature selection, instead using all available features on a small set of data. Their best results are achieved by an SVM with a quadratic kernel. On average, an accuracy of 99.60%, a recall rate of 99.01% and an F_1 score of 0.9935 were reported, outclassing an SVN with a linear/cubic kernel and KNN/DT classification methods.

CIC-DDoS2019

Given the constant change in how DDoS attacks operate, datasets age quickly and shortcomings often arise. A new work by the CIC analysed older DDoS datasets and identified key issues. They propose a new taxonomy for DDoS attacks and generate a new DDoS dataset with pcap files and netflows that remedy these issues. The dataset is split over two separate days and contains multiple attacks, including exploitation (SYN/UDP Flood, UDP-Lag) and reflection (MSSQL, DNS, PORTMAP, TFTP, and more) attacks. Table 4.5 shows the attacks present per day. There are non-overlapping attacks in both sets. For instance, the testing day contains PortMap attacks, yet this attack was not executed during the training day. Moreover, the training day contains six attacks that are not present in the testing set, of which the most-occurring attack (TFTP) does not occur in the testing set. The labelled flows were generated using their CICFlowMeter tool. Their infrastructure included a single Ubuntu server, a Fortinet firewall and four Windows workstations running different version of Windows.

Training day (1 Dec.)	Testing day (3 Nov.)
NTP	PortMap
DNS	NetBIOS
LDAP	LDAP
MSSQL	MSSQL
NetBIOS	UDP Flood
SNMP	UDP-Lag
SSDP	SYN Flood
UDP Flood	
UDP-Lag	
WebDDoS	
SYN Flood	
TFTP	

Table 4.5: Attacks used in CIC-DDoS2019.

Elsayed et al. [8] designed a deep learning method using a recurrent neural network (RNN) with an auto-encoder. They remove socket information to prevent overfitting, remove rows with missing or infinite values and normalize the data for input to their RNN, ending up with 77 input features. A train/test split of 70/30% was found to be ideal. Furthermore, the total number of samples per split was altered such that the distributions of possible attack types were balanced. With a low learning rate of 0.0001, they achieve an accuracy and recall of 99%. Moreover, the specificity is 99%, as well, outperforming traditional ML methods such as DT, SVM, logistic regression and others.

[19] constructed a light-weight decision tree for a CAIDA dataset in 2007, CIC-DoS2017 and CIC-DDoS2019 with only three features (mean packets per seconds, forward bytes and forward IAT mean, all per flow) that can classify with an accuracy of at least 99.69%. For CIC-DDoS2019, the accuracy was 99.93%. Moreover, the misclassification rate for both benign and malicious flows was 0.3% for CIC-DoS2017 and 0.1% for the other datasets. Their feature selection process was based on low variance filtering and a threshold that excluded certain features. These features then helped the sampling process, only selecting 24000 flow records for classification on the CIC-DDoS2019 dataset. Moreover, their decision tree uses information gain as quality measure, and after pruning has a tree depth of 4 with a minimum leaf node size of thirty samples.

4.1.4. inSDN

A new rising domain concerns the network type of an organization's infrastructure. Software-Defined Networks are increasing in popularity due to the separation of the control plane from the data plane, resulting in easier network management for administrators. Previous studies on creating SDN-styled datasets focus only on one aspect of SDNs and suffer from unideal attack variety. [9] worked on a testbed environment to generate datasets that address these issues. The main differences are found in the attack variety, which was analysed for SDNs, and the inclusion of popular application services as normal traffic. Their dataset includes seven attack types, including (D)DoS, web attacks, authentication bruteforcing (R2L), botnet attacks (Malware), network discovery (Probe) and using software exploits for privilege escalation (U2R). A subset of these attacks are directed to a separate Metasploit victim server as they contain the necessary infrastructure for the attacks.

The data is split in three separate .csv files, based on the traffic type and target machines. The first group contains benign traffic only. The second group contains the captured traffic towards the Metasploit victim server, whereas the third group represents the attack records inside the Open vSwitch (OVS) server.

In a different paper by the same researchers [29], they explored a classification method using a LSTM-based auto-encoder. All socket information (flow ID, IP, etc.) is removed from the feature list to prevent overfitting on the data. Furthermore, they randomly selected samples from the dataset (about 25%) to reduce the computation time. Using a simple fixed threshold for the l_2-norm error, where a low error indicates normal traffic and high error anomalous traffic, the results are far from ideal. An accuracy of only 74.1% and an F_1 score of 0.825 is reached, both mostly affected by the low precision of 0.7111, indicating a relatively high amount of false positives. To solve this, an One Class SVM with radial basic function kernel is added to the workflow. On its own, it achieves a better accuracy value of 87.5%, F_1 score of 0.91 with a greatly increased precision of 0.89. When combined with their existing LSTM-Auto-encoder, the precision further increases to 0.93, with corresponding improvements to F_1 score and accuracy as well.

4.2. Related Work

In this section, we show relevant work by other researchers that help us determine useful features to use in our generalized classification models.

4.2.1. Statistical features

Anomaly detection for network traffic knows a wide variety of techniques. Starting with simple profiling or binning, where data points are simply filtered and grouped based on some criteria, to traditional machine learning methods such as SVM, DT, RF, et cetera, to even more complex methods such as neural network implementations.

[3] performed dimensionality reduction for ML-based IoT botnet detection. They aggregated network traffic in 5 different categories. Based on host IP, host MAC and IP, src/dest IP combination, src/dest IP/port combination, and finally network jitter, which they define as the time interval between packets when aggregated by src/dest IP. Their features revolved around statistical data of packet count in time

windows depending on the category. The categories were tested on decision tree and k-NN. Decision tree outperformed k-NN with an average accuracy of 0.9863 over multiple feature combinations.

[7] did research on DDoS protection for consumer IoT devices. Their assumption was that IoT network behaviour is very limited in terms of actors and should easily be mapped because of known behaviour. They split the features in two sections: stateless and stateful. Stateless features are light-weight, flow-independent features such as packet size, inter-packet arrival time and protocol. Stateful features are context-dependent and are represented as aggregated netflows by source IP and non-overlapping time windows. Examples are the bandwidth (average bytes over time) and the unique amount of destination IP addresses. All of their techniques, including k-NN, Linear SVM, Decision Tree (with Gini score), Random Forest (with Gini score) and a 4-layer fully-connected neural network had great F1 scores around 0.997. Furthermore, they calculated the Gini score for all the features and found that the stateless features are the most predictive, with packet size being the most important.

[19] worked on a light-weight solution for (D)DoS attacks. Using flow-based data from a CAIDA dataset in 2007, CIC-DoS2017 and CIC-DDoS2019, they constructed a light-weight decision tree with three, five and seven statistical features. Starting with 40 features, they measured the variance caused by each feature and only included features above a certain threshold. For example, they used the mean time between two packets sent in the forward direction received by the target, the number of bytes in the forward direction received by the target, the average packet size of a flow, the number of forward packets received by the target, and others. The authors opted for a decision tree with the C4.5 algorithm, which is a revised version of the Iterative Dichotomiser 3 (ID3) algorithm, which makes use of the information gain ratio for the selection of attributes with the best split. For all three datasets, the algorithm performs the best with three features, with an accuracy over 99.69%. Moreover, the misclassification rate is very low, with CIC-DoS2017 having a rate of 0.3% for both false positives and false negatives. The other datasets record a misclassification rate of 0.1%.

4.2.2. Temporal features

[38] performed a cluster analysis on CTU-13 netflow data for botnet detection. Their method utilized temporal and spatial features found in network traffic data, referenced from [42]. A sliding window of one minute was used to split the netflows in intervals. The netflows were then aggregated according to source IPs, transforming the analysis to host-level classification. A clustering algorithm was then applied with two clusters, 1000 iterations and six statistical features, including the unique number of source and destination ports, destination IP addresses, and the number of netflows, bytes and packets. An accuracy of 96% was achieved on classifying benign vs. botnet aggregated flows, but 27.7% of actual botnet aggregated flows were misclassified, illustrating the class imbalance of this dataset.
[25] provided a feature set analysis in 2019 for network traffic classification. They experimented with burstiness features (i.e. inter-packet idle time, an extension of inter-packet arrival time) and periods of inactivity. They define a burst by a maximum idle time between packets. Once that maximum time has been exceeded, the previous packets that had a lower inter-packet idle time will be grouped as a single burst. They trained a C5.0 decision tree and generated traffic from 11 different applications by 10 users. An accuracy of 98% was achieved with high corresponding sensitivity and specificity per application. A more in-depth analysis on ideal burst time or idle threshold was not available.

4.3. Experiment Outline

This section will present the methodology behind the classification pipeline. The goal is to design a general model that can accurately separate malicious traffic from benign traffic over multiple datasets. First, we will describe each component of the classification pipeline. We then present the results in the form of confusion matrices and bar charts, comparing them to prior work when available, and finally reflect on the achieved classification score.

A visual representation of the classification pipeline can be seen in Figure 4.4. The classification pipeline has three components through which the input will run. First, data preprocessing is required to obtain a homogeneous representation of the data from each dataset, such that we can freely explore it and select features from. Once the data is converted to a common format, feature selection takes place through analysis and insights from earlier works, as described in Section 4.3.2. More specifically, for connection-level datasets we are interested in testing three feature sets. A general feature set that focuses on common features, such as protocol, amount of packets in a flow, amount of bytes in a flow,

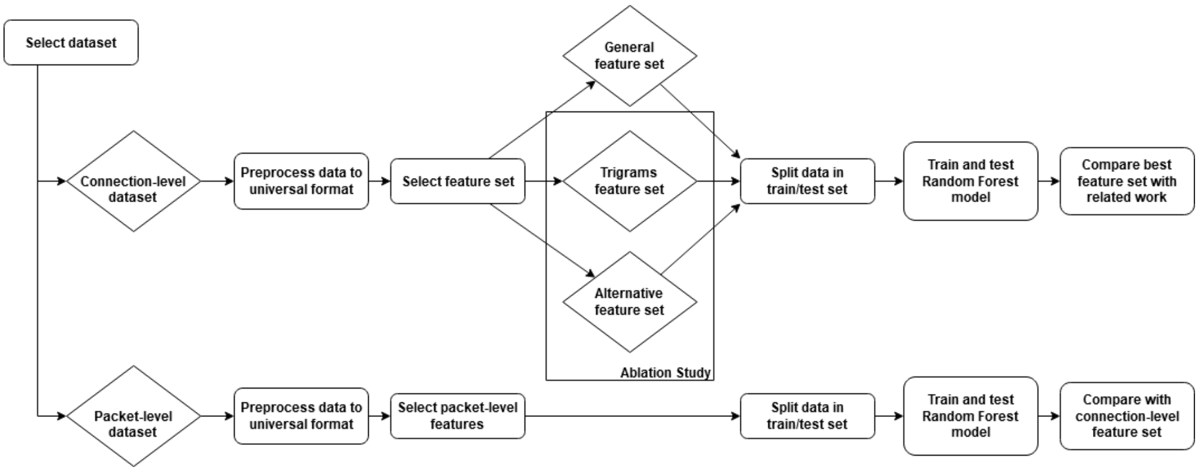

Select dataset
Connection-level dataset
Preprocess data to universal format
Select feature set
General feature set
Trigrams feature set
Alternative feature set
Ablation Study
Split data in train/test set
Train and test Random Forest model
Compare best feature set with related work
Packet-level dataset
Preprocess data to universal format
Select packet-level features
Split data in train/test set
Train and test Random Forest model
Compare with connection-level feature set

Figure 4.4: High-level overview of the classification pipeline. A separate process is taken based on the level of data used. In both cases, we use a Random Forest model for binary classification.

et cetera; An alternative feature set that explores more statistical properties of the dataset and a transformed version of the alternative feature set that uses the mean and standard deviation of each feature as trigrams in network intrusion detection. These last two feature sets form an ablation study to measure the effectiveness of trigrams. The next component concerns the classification model. A Random Forest implementation was chosen to perform the classification experiments. This component is satisfied when most, if not all, datasets can be classified correctly at a reasonable degree. Finally, Section 4.3.4 shortly presents the evaluation metrics to be reported for each dataset. For each experiment, we report the accuracy, recall and specificity of the binary classification results. Even though accuracy is not a great quality metric in this domain due to class imbalance, it often is reported by related work, sometimes as the only metric available. Recall and specificity, also known as true positive and negative rate, respectively, do work well for imbalanced datasets. A large class imbalance towards benign behaviour does not hide the malicious classification results, as we can calculate how many actual malicious instances are classified correctly without needing information for benign instances.

For the experiments, we used Python 3.6.9 with common big data and machine learning libraries, including Pandas[4], NumPy[5] and scikit-learn[6]. For reproducibility of the results, we share the Jupyter notebooks online on a Github repository[7] under the research group this thesis has been researched for.

4.3.1. Data preprocessing

In Section 4.1, we defined six datasets that were used. Most of these datasets have different formats, e.g. different header names and delimiters. Some of them also have features that others do not have, therefore we transformed each dataset to a universal format and removed features that are not globally available. Moreover, some datasets come in different levels of data, as well. All datasets have connection-level data, but one carries packet-level data suitable for experiments, too. Thus, we will need to make a distinction between the two types of data level representations in terms of available features.

In general, the label classes were renamed to a more consistent format for display purposes. Furthermore, we used stratified 5-fold non-shuffled validation with a 80/20 train/test split, grouping the data points based on the attack type (multiclass label) and using chronological ordering to respect the sequential property. In all cases, we prepared the code for binary classification. In the case that a dataset provides its own train/test split, we used it instead, if applicable. Finally, if a dataset contains categorical columns that we used, such as the protocol type, we converted that column to a one-hot-encoded version, as the chosen classification method described in Section 4.3.3 only works with numerical fea-

[4]`https://pandas.pydata.org/`
[5]`https://numpy.org/`
[6]`https://scikit-learn.org/stable/`
[7]`https://github.com/tudelft-cda-lab`, repository is called "NIDS Datasets Quality Evaluation"

tures.

CIDDS-001
The network traffic is split in eight separate .csv files based on week number and whether the data is from the internal or external network. Table 4.3 provides an overview of the attacks that happen per platform and week. Due to the traffic differences between the weeks and platforms, we divided the dataset in two separate dataframes as a train and test split. For both the internal and external network, week 1 and 3 correspond to the training set, while week 2 and 4 are the testing set. The train/test traces are concatenated based on platform followed by week number. As example, for the train split, the external platform data of week 1 and 3 is concatenated with the internal platform data of week 1 and 3. These two resulting dataframes then serve as input to the rest of the pipeline.

UNSW NB15
The train and test split supplied by the authors require no preprocessing besides the general remarks mentioned at the start of this section. This dataset contains the most unique protocols out of all the datasets, with a staggering 176 unique protocols present. This, in turn, will inflate the feature list size of this dataset. Since we wish to generalize the feature set for the pipeline, we did not remove any protocols from the one-hot-encoded version.

inSDN
Since the data is split in three separate files based on traffic type, we simply concatenated the read-in dataframes to combine the data. The train/test split was done such that the class ratios remained the same in both sets. Finally, the data did not contain binary labels yet, so we applied the benign class label to the benign traffic trace and the malicious class label to the other traces.

CICIDS2017
Each of the five days have separate traces and are further split based on the time of day during the data capture. In total there are eight traces. We combined all traces in one dataframe by concatenating them. The dataset suffers from NaN values in arbitrary columns, so we dropped any rows that contain NaN values. A binary label class was also added based on the multiclass label values.

CICIDS2018
The traffic data was generated per day. For each day, a subset of attacks were executed on the live network. Some attacks were only executed once or only exist on one trace, therefore we concatenated all flow-based .csv files in one large dataframe. The binary label class was again added. Due to the size (over 6GB of raw data), we only select a subset of features to be parsed into the dataframe. CICIDS2018 also suffers slightly from NaN values, so any affected rows were removed.

CICIDS-DDoS2019
This dataset has the trace files split per attack and already provides a train and test split based on two separate capture sessions. We concatenated all files that are in the intersection of all attacks for the two days (see Table 4.5), meaning we included MSSQL, NetBIOS, UDP Flood, UDP Lag, LDAP and SYN Flood DDoS attacks. A few hundred flows are marked as WebDDoS, which we also removed since there is too little data to learn and classify on it. Although the table refers to the trace of 3 November as the testing day and 1 December as the training day, we switched the sets just before training the model, since the classification score was higher with 3 November as the training set and 1 December as the testing set. We suspect this has to do with the increased sample size of most classes. We reduced the memory footprint by only parsing selected features and explicitly typesetting data types. After, the binary label class is added based on the traffic type per flow.

For the packet-level data, we work with *tshark*. To be able to work with the immense amount of data, we only process the testing set and throw away the second half of traffic during each attack period. *tshark* is unable to parse all rows perfectly, so rows with invalid column values are removed as well. Furthermore, any packets with the protocol different from ICMP, IGMP, TCP or UDP are discarded, as we are not interested in smaller protocols for DDoS anomaly detection. Port values are combined, since the parsed values are done per protocol. Combining them to a source and destination port column works best for classification purposes. ICMP and IGMP are allocated -1 values for their ports to match the flow-based representation. Labelling of each packet is done by comparing packets to their summarized version in the flow-based dataset. Out of 39 million packets, we discard nearly 10% of packets since they could not be labelled due to timestamp mismatches, ending up with about 35.7 million packets, with 35 million being malicious.

4.3.2. Feature selection

Once we have the data in a homogeneous representation, we can start the feature selection process. Our primary focus will be on connection-level data, as that representation contains the most prior work and has the most available datasets.

Due to the sequential property of network traffic, temporal features such as the inter-packet arrival time, flow duration and some statistical properties about all packets within a flow are shown be important, as proven in [19, 25, 38]. To make additional use of the sequential property, we explore a feature set focused on N-grams, i.e. calculating features on dataset rows within a sliding window of length n. We settled on $n = 3$, also known as trigrams, with a forward sliding window. For each flow grouped by their source IP address and sorted chronologically, if there are at least two more flows after that data point, we calculate the mean and standard deviation of several statistical features. For aggregated flows that do not meet this condition, thus for which no new features are calculated, they are removed from the dataset for the trigrams classification experiments. Furthermore, for CIC-DDoS2019 there is too much data to fit into memory while also creating new trigrams features, even when throwing away the unused classes. To solve this issue, we only retain the first 60% of each class label in the original training set, discarding the rest. Finally, to measure the impact of using trigrams on the classification score, we perform an ablation study using the raw feature set of Table 4.6b for every flow instance. The difference between this alternative feature set and the trigrams feature set will illustrate the effect of trigrams on these datasets.

Besides temporal features, it is well-known that standard features such as protocol, packet size and certain flags form a healthy basis for detecting anomalous behaviours. In some contexts, for example IoT networks, network ports are also a great indicator, as most communication between devices has limited variety and is relatively consistent due to the low variety in data being sent. Since we have a large amount of datasets, it is infeasible to optimize a method for each dataset. Thus, our work will be based on the common features between datasets and their respective prior work as defined in the chapters about related work and data.

For connection-level data, we settled on the features listed in Table 4.6. For the general feature set, NB15 includes high-level custom features based on a combination of other features and corresponding threshold values, such as `ct_state_ttl`, which gives the count of packets within a specific range of TTL value per state. The authors do not provide details on the used value ranges per state for this feature. Interestingly, during initial testing we found this feature to explain a high amount of variance within the data, and therefore we included it in this specific dataset.

For the trigrams feature set, we selected similar statistical features, but expanded upon them with bytes and amount of packets per second. Moreover, we used additional information from inter-packet arrival times as related work found this statistic to work well in classifying traffic, resulting in a feature set size of 30. Figure 4.5 gives a concrete example of how the data looks like for the three connection-level feature sets.

With no related work available for packet-level data, we also set out to do some preliminary analysis on the class/protocol distributions of certain features to see if they are (abnormally) good indicators of malicious behaviour. Figure 4.6 illustrates the exploration of the frame length and time to live features. Looking at the overlap between protocol type and class label in Figure 4.6a, we see that most malicious behaviour happens with UDP traffic and is between 300 and 1500 bytes long. Inspecting Figure 4.6b in the same manner, we find that TCP is more prevalent in malicious behaviour than before, indicating that the malicious traffic over TCP prefers some specific TTL values. Noteworthy are the large spikes in both figures, which indicate a specific attack or service being repeated many times.

Applying a simple binning technique already resulted in a TPR of 99.9% and a TNR of 77%. Discretizing frame TTL and frame length was based on the protocol histograms, as each protocol had a clearly different distribution of the respective values. Since binning uses 'nice' value thresholds, such as the mean or certain quantiles, the bins remain general and implicitly do not work well with spikes, as seen in Figure 4.6. Therefore, we will apply a machine learning technique to optimize the model and improve the TNR. Nonetheless, the features are promising enough to continue using.

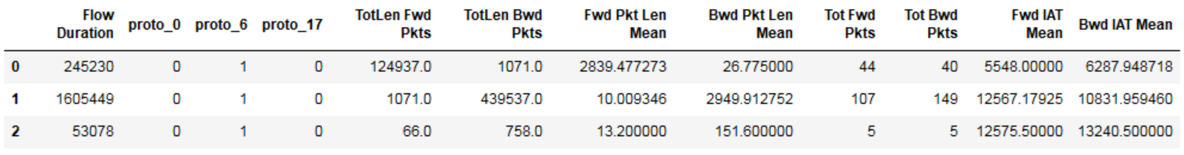
Dataset	Features
CIDDS	Flow duration Src/Dst Bytes per flow (forward and backward) Src/Dst Packets per flow (forward and backward) Mean size of packets per flow Protocol (One-hot encoded)
CICIDS2017/2018/ CICDDoS2019/inSDN	Mean inter-packet arrival time (forward and backward)
UNSW-NB15	Src mean TTL ct_state_ttl

(a) General feature set.

Dataset	Features
CIDDS	Flow duration Src/Dst Bytes per flow (forward and backward) Src/Dst Packets per flow (forward and backward) Mean size of packets per flow
CICIDS2017/ CICDDoS2019/inSDN	Src/Dst Bytes min/max per flow (forward and backward) Flow bytes/s Flow packets/s Flow Std. Dev. inter-packet arrival time Flow Min/max inter-packet arrival time Flow mean inter-packet arrival time

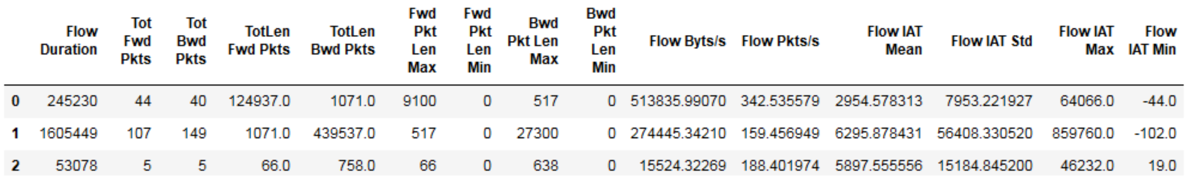
(b) Alternative feature set. For the trigrams feature set, the mean and standard deviation for every feature is calculated over a sliding window of three flows.

Table 4.6: Features used for connection-level classification. Datasets contain all listed features up and till mentioned position.



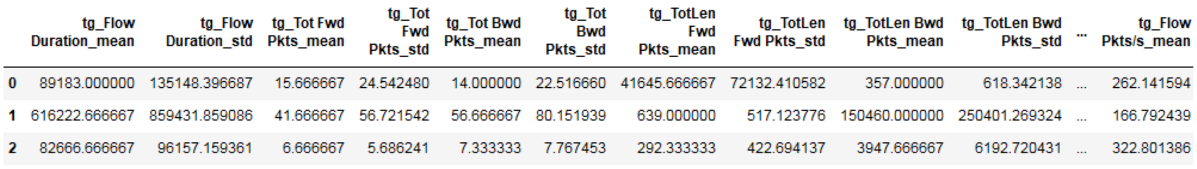
	Flow Duration	proto_0	proto_6	proto_17	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Mean	Bwd Pkt Len Mean	Tot Fwd Pkts	Tot Bwd Pkts	Fwd IAT Mean	Bwd IAT Mean
0	245230	0	1	0	124937.0	1071.0	2839.477273	26.775000	44	40	5548.00000	6287.948718
1	1605449	0	1	0	1071.0	439537.0	10.009346	2949.912752	107	149	12567.17925	10831.959460
2	53078	0	1	0	66.0	758.0	13.200000	151.600000	5	5	12575.50000	13240.500000

(a) Three netflows of inSDN with the general feature set.



	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Bwd Pkt Len Max	Bwd Pkt Len Min	Flow Byts/s	Flow Pkts/s	Flow IAT Mean	Flow IAT Std	Flow IAT Max	Flow IAT Min
0	245230	44	40	124937.0	1071.0	9100	0	517	0	513835.99070	342.535579	2954.578313	7953.221927	64066.0	-44.0
1	1605449	107	149	1071.0	439537.0	517	0	27300	0	274445.34210	159.456949	6295.878431	56408.330520	859760.0	-102.0
2	53078	5	5	66.0	758.0	66	0	638	0	15524.32269	188.401974	5897.555556	15184.845200	46232.0	19.0

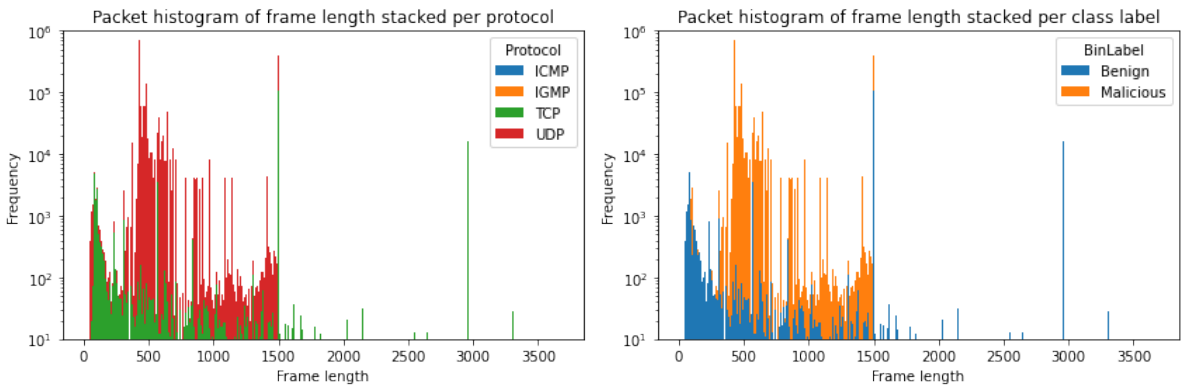
(b) Three netflows of inSDN with the alternative feature set.



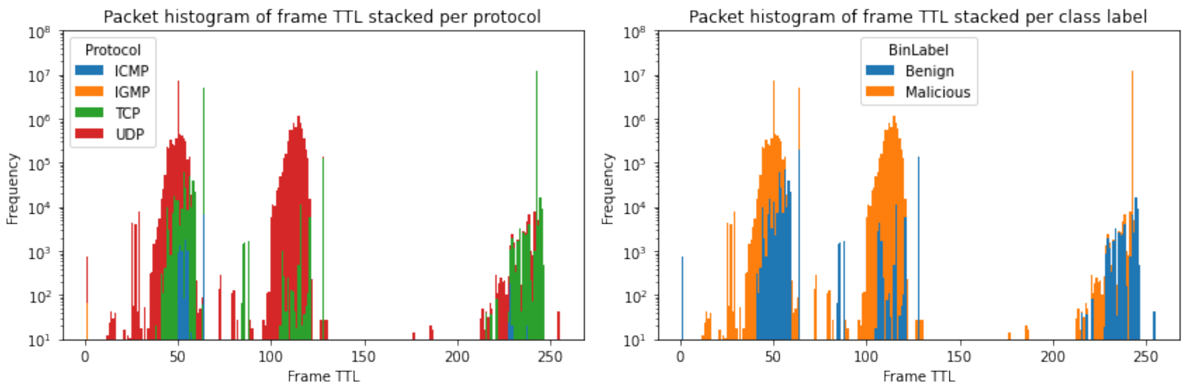
	tg_Flow Duration_mean	tg_Flow Duration_std	tg_Tot Fwd Pkts_mean	tg_Tot Fwd Pkts_std	tg_Tot Bwd Pkts_mean	tg_Tot Bwd Pkts_std	tg_TotLen Fwd Pkts_mean	tg_TotLen Fwd Pkts_std	tg_TotLen Bwd Pkts_mean	tg_TotLen Bwd Pkts_std	...	tg_Flow Pkts/s_mean
0	89183.000000	135148.396687	15.666667	24.542480	14.000000	22.516660	41645.666667	72132.410582	357.000000	618.342138	...	262.141594
1	616222.666667	859431.859086	41.666667	56.721542	56.666667	80.151939	639.000000	517.123776	150460.000000	250401.269324	...	166.792439
2	82666.666667	96157.159361	6.666667	5.686241	7.333333	7.767453	292.333333	422.694137	3947.666667	6192.720431	...	322.801386

(c) Three netflows of inSDN with the trigram feature set. Most features are hidden due to illustration space.

Figure 4.5: Connection-level feature list example for inSDN. Label per data instance is omitted.



(a) Frame length (bytes) exploration per protocol or class label



(b) Frame TTL exploration per protocol or class label

Figure 4.6: Data exploration of frame length and time to live features on packet-level data. Y-axis is in log scale.

Dataset	Features
	Frame length
	Frame TTL (Time To Live)
	Protocol (One-hot encoded)
	TCP flags (One-hot encoded)
	SYN/ACK/FIN/packet count per source host
	Discretized Frame TTL
CICDDoS2019	Discretized Frame length

Table 4.7: Features used for packet-level classification.

We settled on the features listed in Table 4.7. Although packet-level data does not carry the benefits of netflows (single packet versus an entire connection), it does carry more individual information per packet. Since DDoS2019 contain several DDoS attack vectors on TCP, one of them being SYN flood, classifying with TCP flags is very promising.

4.3.3. Classification method

With the available features in mind, we designed a method that works well on both data levels. For every data level, the aim is to achieve a good overall result on as most datasets as possible. With this result, we can answer our research question, given the used feature sets. A random forest provides an aggregated decision process that is relatively easy to understand for human analysts. Therefore, network data and white-box techniques such as random forests pair well in analysis. For this reason, we used a random forest implementation in our classification pipeline for connection- and packet-level datasets. Given the success of random forest implementations on network intrusion detection, especially when compared to other traditional machine learning methods such as SVM, k-NN, et cetera, it is a great contender to compare with current state-of-the-art solutions. For both connection-level and packet-level classification, we use the scikit-learn out-of-the-box Random Forest classifier with validated hyperparameters, to illustrate the simplicity of classification on the datasets.

The goal of validating/finding the best hyperparameters for a machine learning model is to increase the fitness of the model on the data. The fitness in this problem context is defined by the reported classification score metrics. A Random Forest has many hyperparameters to test - some focused on the forest itself, some on the individual decision trees in the forest. It is infeasible to test all hyperparameters as the time needed for the search doubles with every new hyperparameter value, so we applied default values to some hyperparameters that we deemed less important than others. Since random forests control overfitting by design, as opposed to decision trees, our implementation did not set a max depth that a tree can reach. This means that each tree will continue to expand until all leaves are pure or contain less than the minimum amount of samples needed to split. By default, the minimum split size is set to 2, meaning a node only needs two samples to create a new split. Since two samples in network data generally do not represent the data well enough for a separate decision to be created to split them, we set the minimum split size slightly higher to 10 samples and adjusted the minimum sample size for a node required to be a leaf to half that amount, to further smoothen the model.

For the hyperparameters that we do deem to be important, we perform a grid search to find the best combination of values. The amount of trees in a forest and the maximum ratio of data each tree can use for learning were determined by a hyperparameter grid search, together with the optimal quality measure for network anomaly detection. Each parameter set was evaluated by reporting the balanced accuracy metric of the model on the binary classification problem. Training was done on the train set, whereas we evaluated the chosen parameters on the validation set. We split the data in a 60/20/20 train, validation and test set, ignoring the test set to prevent optimising on the dataset - leading to overfitting. We tested the forest size n with the values {1,5,10,25,50,100} as initial experiments indicated that higher forest sizes did not bring significant performance increases. We also tested the ratio of data per tree with the values {0.7,0.8,0.9}, and Gini impurity vs information gain for the quality measure resulting in 32 iterations per fold.

Performing this practical analysis on the datasets will provide valuable insight to the complexity of the data and effectiveness of corresponding feature sets, as white-box machine learning models contain a decision making process that is more easily interpretable by human operators for further analysis. Moreover, with this information, we could also draw conclusions on the, perhaps, unnecessary complexity of solutions accompanying the datasets, as described in Section 4.1. We illustrate the findings of this grid search per data level in the two paragraphs below.

Connection-level RF

For the hyperparameter grid search experiment, we chose the CIDDS and CICIDS2017 datasets to reduce the time needed to perform the grid search and use the general feature set. Table 4.8 presents the findings for the hyperparameter grid search that focused on random forest size, ratio of data to give each tree in the forest and the quality measure to be used. From this table, it becomes immediately clear that a higher forest size in nearly all cases lead to a higher balanced accuracy. However, there was no significant increase after a size of 25 for these two datasets. Still, with implicit reduction of overfitting in mind, we continued with the default forest size of 100. There is no clear difference between Gini

	CIDDS-001						**CICIDS2017**					
	Gini Impurity			Information Gain			Gini Impurity			Information Gain		
n	0.7	0.8	0.9	0.7	0.8	0.9	0.7	0.8	0.9	0.7	0.8	0.9
1	0.9866	0.9406	0.9871	0.9864	0.9820	0.9697	0.7707	0.8201	0.7316	0.7326	0.7951	0.7377
5	0.9833	0.9893	0.9775	0.9866	0.9902	0.9793	0.7712	0.7607	0.7586	0.7584	0.7587	0.7588
10	0.9874	0.9853	0.9869	0.9916	0.9837	0.9898	0.7598	0.7611	0.7606	0.7589	0.7603	0.7568
25	0.9871	0.9885	0.9890	0.9864	0.9876	0.9906	0.7608	0.7604	0.7610	0.7601	0.7589	0.7600
50	0.9892	0.9864	0.9889	0.9898	0.9887	0.9893	0.7603	0.7605	0.7595	0.7594	0.7598	0.7598
100	0.9890	0.9900	0.9869	0.9893	0.9898	0.9895	0.7611	0.7616	0.7607	0.7603	0.7600	0.7603

Table 4.8: Random Forest hyperparameter grid search scores for the connection-level model. Reported values are balanced accuracy scores averaged over 5 folds. Each column represents a combination of the objective function and data ratio per dataset. The rows coincide with the tested random forest sizes.

	CICDDoS2019					
	Gini Impurity			Information Gain		
n	0.7	0.8	0.9	0.7	0.8	0.9
1	0.99823	0.99835	0.99836	0.99829	0.99824	0.99829
5	0.99836	0.99836	0.99837	0.99838	0.99837	0.99829
10	0.99838	0.99838	0.99837	0.99836	0.99838	0.99837
25	0.99838	0.99837	0.99838	0.99838	0.99837	0.99837
50	0.99838	0.99838	0.99837	0.99838	0.99837	0.99837
100	0.99838	0.99837	0.99837	0.99838	0.99838	0.99837

Table 4.9: Random Forest hyperparameter grid search scores for the packet-level model. Reported values are balanced accuracy scores averaged over 5 folds. Each column represents a combination of the objective function and data ratio per dataset. The rows coincide with the tested random forest sizes.

impurity and Information Gain / entropy. In most cases, Gini impurity scored higher with a very slim margin. We also continued with the default choice here, meaning we took Gini impurity as the objective function. Finally, the ratio of data per tree had little variation as well. One reason for this might be the vast amount of data, neatly split and ordered in a stratified 5-fold fashion. Taking into account the selected parameters so far, a data ratio of 0.8 outperforms the other ratios slightly. Thus, our random forest experiments will use a forest size of 100, Gini impurity as the objective function, and a data ratio of 0.8 per tree.

Packet-level RF

Since we only have access to one labelled packet-level dataset, our hyperparameter grid search only ran on that one dataset - CIC-DDoS2019. Table 4.9 contains the balanced accuracy scores averaged over all folds, per hyperparameter setting. It immediately becomes clear that nearly all hyperparameter combinations are optimal. The almost identical results are most likely explained by the abundance of data in every fold. The random forest does not need to be complex in order to classify well. Instead, a decision tree would fare only slightly worse than an ensemble of them. We continued with the same hyperparameters as the connection-level datasets for the same reasons, as well, where $n = 100$, the data ratio is 0.8 and the Gini impurity is used as the information criterion.

4.3.4. Evaluation metrics

As explained in Section 2.4, the results of classification can best be represented in a confusion matrix. For each dataset, we present the classification results per fold or iteration, and additionally provide the mean and standard deviation over all folds. The comparison with previous literature will focus on binary classification and use the best feature set on the data level. More specifically, this means that we report the accuracy, recall and specificity of our work and related work, whenever available.

4.4. Results

In this section, we provide an overview of results obtained from our random forest implementations on the selected datasets. First, we focus on the random forest implementation that deals with connection-level datasets. We inspect the results on all three feature sets, provide feature importance scores and metrics such as accuracy, recall and specificity, and relate them to previous work. The packet-level implementation on CIC-DDoS2019 follows in the same manner, but without the comparison to previous work with broader metrics. Finally, we conclude the chapter in Section 4.5 and answer our research questions.

4.4.1. Connection-level RF

Table 4.10 shows the contingency tables per dataset evaluated with our random forest implementation on the general feature set. For four out of six datasets, we performed stratified 5-fold non-shuffled cross-validation, where each iteration presents a different test fold consisting of 20% of the data. For the other two datasets, five iterations of the same data were used. For CICIDS2017, CSE-CIC-IDS2018 and inSDN it is noteworthy to mention the standard deviation for false positives and false negatives through the folds.

	UNSW-NB15*				CIDDS-001				CICIDS2017			
Iter.	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	104821	53909	2091	14520	331882	5632485	4745	1891	106142	451846	2692	5197
2	104705	53988	2012	14636	332979	5633486	3743	794	104798	452395	2143	6541
3	104736	53963	2037	14605	331479	5632462	4767	2294	108120	447615	6923	3219
4	105219	53878	2122	14122	331935	5632478	4751	1838	108421	440648	13889	2919
5	104940	53965	2035	14401	330436	5633220	4009	3337	98088	452526	2011	13252
Mean	104884.2	53940.6	2059.4	14456.8	331742.2	5632826.2	4403	2030.8	105113.8	449006	5531.6	6225.6
σ	$\pm 0.18\%$	$\pm 0.08\%$	$\pm 1.97\%$	$\pm 1.29\%$	$\pm 0.24\%$	$\pm < 0.01\%$	$\pm 9.96\%$	$\pm 40.38\%$	$\pm 3.57\%$	$\pm 1.01\%$	$\pm 82.34\%$	$\pm 60.32\%$

	CSE-CIC-IDS2018				CIC-DDoS2019*				InSDN			
Iter.	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
1	502638	2668262	28677	47009	13641278	11352	105	1663578	50241	13138	547	4853
2	500641	2675731	21208	49006	13668445	11417	40	1636411	54889	13264	421	205
3	504736	2677007	19932	44911	13659391	11400	57	1645465	54920	12824	861	173
4	501215	2652894	44045	48432	13659373	11375	82	1645483	53192	12940	745	1900
5	506367	2595222	101716	43280	13659391	11379	78	1645465	54629	13008	676	463
Mean	503119.4	2653823.2	43115.6	46527.6	13657575.6	11384.6	72.4	1647280.4	53574.2	13034.8	650	1518.8
σ	$\pm 0.43\%$	$\pm 1.15\%$	$\pm 70.81\%$	$\pm 4.63\%$	$\pm < 0.06\%$	$\pm 0.20\%$	$\pm 30.72\%$	$\pm 0.54\%$	$\pm 3.33\%$	$\pm 1.18\%$	$\pm 23.56\%$	$\pm 117.46\%$

Table 4.10: Binary contingency tables for each dataset with the general feature set using the Random Forest implementation on connection-level data. *These datasets already had an explicit train/test split and therefore performed five iterations on the same data, instead of on five separate folds.

	CIDDS-001				CICIDS2017			
Iter.	TP	TN	FP	FN	TP	TN	FP	FN
1	325281	5613675	16835	8490	110421	447551	1	833
2	320660	5578904	51605	13112	110636	447369	183	618
3	322954	5568207	62302	10818	110371	447256	296	883
4	318070	5630370	139	15702	110832	446533	1018	423
5	314418	5630417	92	19354	109168	446893	658	2087
Mean	320276.6	5604314.6	26194.6	13495.2	110285.6	447120.4	431.2	968.8
σ	$\pm 1.18\%$	$\pm 0.47\%$	$\pm 99.51\%$	$\pm 28.03\%$	$\pm 0.53\%$	$\pm 0.08\%$	$\pm 84.30\%$	$\pm 60.14\%$

	CIC-DDoS2019*				InSDN			
Iter.	TP	TN	FP	FN	TP	TN	FP	FN
1	7491925	32339	16	1078201	30025	13385	5	677
2	7460898	32359	96	1109228	30692	13327	63	9
3	7464789	32348	107	1105337	30696	13326	64	5
4	7459666	32357	104	1110460	29001	13368	21	1700
5	7499426	32357	98	1070700	30692	13386	3	9
Mean	7475340.8	32350.8	104.2	1094785.2	30221.2	13358.4	31.2	480
σ	$\pm 0.23\%$	$\pm 0.02\%$	$\pm 6.82\%$	$\pm 1.54\%$	$\pm 2.19\%$	$\pm 0.20\%$	$\pm 86.87\%$	$\pm 138.08\%$

Table 4.11: Binary contingency tables for the trigrams feature set with Random Forest implementation on connection-level data. *This dataset already had an explicit train/test split and therefore performed five iterations on the same data, instead of on five separate folds.

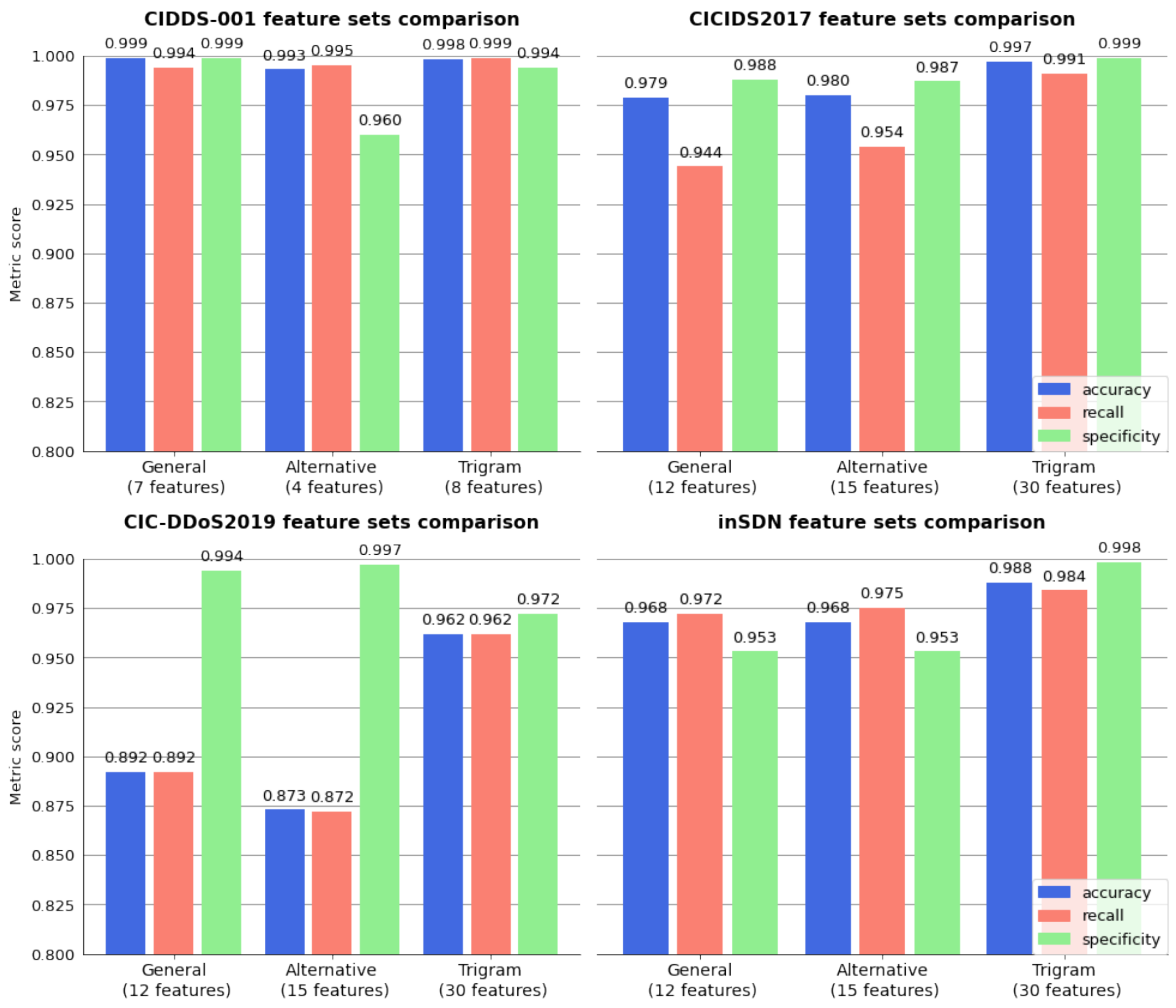


Figure 4.7: Comparison between feature sets on connection-level data using the Random Forest model. Reported classification metrics are accuracy, recall and specificity.

The barplots in Figure 4.7 provide a comparison of the feature set quality on the four compatible datasets, CIDDS-001, CICIDS2017, CIC-DDoS2019 and inSDN. In all cases, the general feature set performs well, with its best result on the CIDDS-001 dataset (99.9% accuracy). For other datasets, however, it performs similar to the alternative feature set and is outclassed by the trigram feature set. The alternative feature set, which we primarily compare to the trigram feature set as part of the ablation study to study the effects of using trigrams vs. normal flow features, is always outclassed by the trigrams feature set, even though they use the same base features. This indicates that trigrams in a connection-level setting provides a positive contribution to the classification quality of the model.

Figure 4.8 contains the metric scores of our best feature set and related work, per connection-level dataset. About half of the papers did not report extensive metrics and therefore could only be compared on one or two metrics, instead of accuracy, recall *and* specificity. For three datasets, our model performed just as well, if not better, than related work. For the other three datasets, the model still performs well (>90% accuracy), but related work found a better solution.

It is interesting to find out where the difference in approach lies versus related work that perform better than us. Starting with UNSW-NB15, the work by Faker et al.[10] managed to extract 41 features from the dataset with their DNN. Although they have a section describing feature importance ranking, they have not shared the results. Moreover, they also did not report the features in the first place. It is, therefore, unknown how they attained a better classification score.

For CIDDS-001, the work by Abdulhammed et al.[1] is marginally better than ours. Their recall is 99.9% instead of our 99.4%. Disregarding the lack of clarity on used feature sets per approach, the researchers claimed that undersampling the majority class, which is benign traffic, to half its original amount slightly benefits the classification results as opposed to keeping the original distribution. In our tests, randomly removing half the majority class instances revealed no increase to classification score. To slightly investigate the difference in feature amount, we ran a separate test with the general feature plus source and destination ports on the undersampled dataset to meet their setup as precisely as possible. The classification scores increased to 99.6% recall, 99.999% specificity and 99.96% accuracy, bringing us very close to the scores of Abdulhammed et al. If their model indeed included socket information, then their better results will most likely not translate well to other environments.

Ullah et al.[41] provide a supposedly perfect binary classification score on CICIDS2017 using only a decision tree. Their feature list includes some other statistical features than the ones we use, mainly focussing on maximum, minimum and standard deviation of feature values within a flow. Most notably, however, is the inclusion of the source and destination port. Their feature selection was based on experiments that maximise the classification score, yet the inclusion of socket information presents unrealistic results. If we include port information in our trigrams feature set for CICIDS2017, we boost our specificity to 99.93%, our recall to 99.64% and our accuracy to 99.87%, indicating a relevant increase in recall rate compared to our trigrams feature set without ports (99.1%). Further investigating revealed that using the exact same features as Ullah et al. resulted in the same classification scores as our trigrams feature set without ports. Their preprocessing lists feature normalisation as another measure, however decision trees (and by extension random forests) are insensitive to transformations as long as the order of values is preserved - which they likely are during standard normalisation. We are, therefore, unable to explain how Ullah et al. obtained perfect classification scores on CICIDS2017.

The result by Kilincer et al.[16] with a high-scoring quadratic SVM model on CSE-CIC-IDS2018 is surprising, because SVMs generally score worse than other machine learning techniques on this type of data. Nonetheless, the authors explicitly stated that they use all 80 features present in the dataset, including destination port numbers. Similar to Ullah et al. in CICIDS2017, the port numbers contributed to the classification score increase. If we add the destination port to the general feature set we used, specificity is increased slightly to 99.09%, recall to 93.55% and accuracy to 98.16%.

Both related work classify very well on CIC-DDoS2019, with the model by Lucky et al.[19] being both simple and nearly perfect in classification. Revisiting the experiment setup of Elsayed et al.[8], they utilized the maximum amount of features, barring columns that represent socket information, such as IP addresses, flow ID, ports and timestamp. Our model only uses 15 base features (30 in total), as opposed to their 77 features on a black-box model. The decision tree by Lucky et al. has one important feature difference with our random forest model. Instead of only focusing on statistics within a flow, one of their three features is the amount of packets per second. Given the repetitive nature of DDoS attacks, especially when they are the only threat present in the dataset, this feature should provide more information than the general features we selected. In a separate experiment, we added this feature to our general feature set. Although it increased the TPR from 89% to about 92%, it still did not reach the score as reported by the researchers. Their sampling method, although not well-described, resampled the data to only 24000 flow records for both training and testing, as opposed to the 50 million present in only the full training set. We suspect this to play a major part in the training and testing performance of the model, but are unable to replicate this due to lack of information given by the researchers.

For each dataset, we provide the top 10 feature importances of the random forest using the general feature set in Table 4.12. The scores are in relation to all features used in classification, and the sum of all scores per feature set amounts to 1. In all cases, it is evident that the length of the packets inside the flows is a great feature for separating malicious from benign netflows, compared to the other tested features. Both the total length within a flow or the length divided by the amount of packets in a flow perform greatly reduced the gini score and thus reduced uncertainty.

Interestingly, CSE-CIC-IDS2018 seems to be the most diverse, with different important features such as the duration of the flow and the inter-packet arrival time explaining more data. Rerunning the experiments on CSE-CIC-IDS2018 with only the top three features reveals that we can get similar results with only the flow duration, forward inter-packet arrival time and the total length of packets in the forward direction of a flow. An accuracy of 95.4% versus 97.2%, a recall of 90.2% as opposed to 91.5% and a specificity of 96.5% instead of 98.4%. A flat reduction on the lower side of 2% per metric. These results illustrate the importance of those three features on CSE-CIC-IDS2018, when compared to the entire general feature set.

Another surprising find was the non-importance of protocol features, with the only exception being

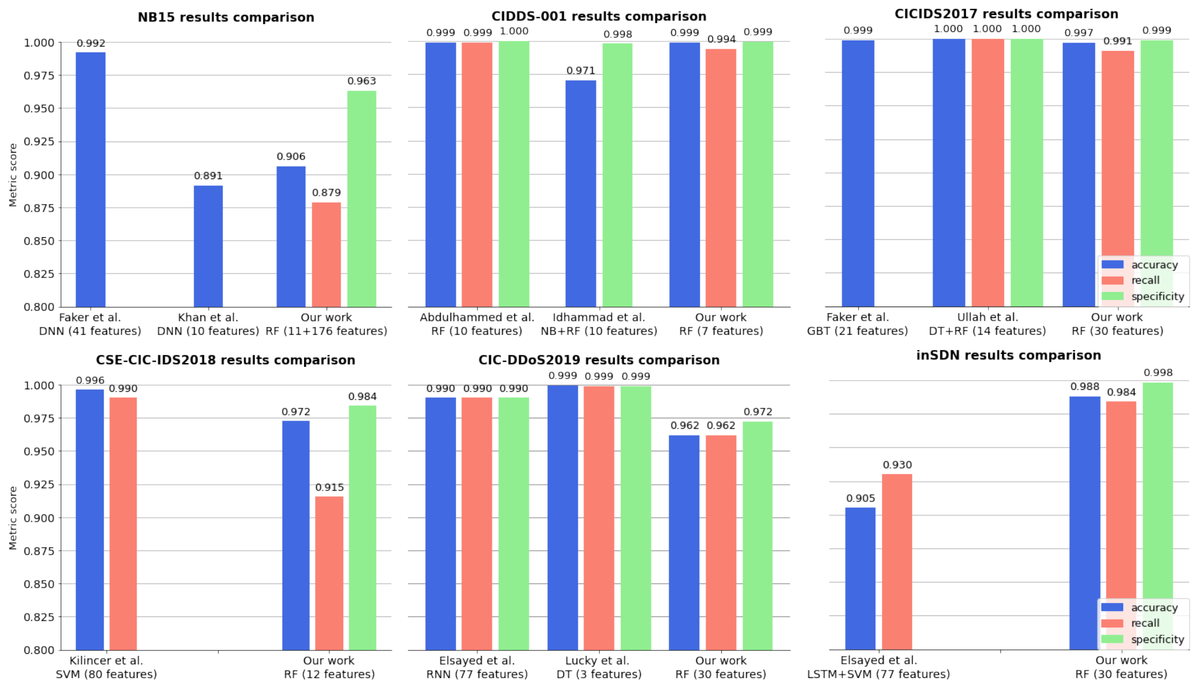


Figure 4.8: Classification score metrics of the best feature set per dataset using the Random Forest implementation, compared to related work.

the UDP protocol in UNSW-NB15. If we rerun the experiment on UNSW-NB15 with only the top three features (sbytes, smean and udp), we still obtain an accuracy of 89.7%, a recall of 87.13% and a specificity of 95.20%. This is only a flat reduction of about 1% for every metric, yet we use only three features with a combined score of 0.4408 instead of 11 general features + 176 one-hot encoded protocol features.

Inspecting the top 10 trigrams feature importances in Table 4.13, we see something similar. For CICIDS2017 and inSDN, The minimum and maximum length of packets in a flow, averaged over three flows and its respective standard deviation, illustrate the importance of the amount of bytes in packets and its aggregated netflow form. Nearly all other features, except the amount of packets and the IAT, do not appear in the top important features.

Connection-level conclusions

With these experiment results, we have shown that classifying well (>90% metric scores) on connection-level datasets with a general classification model is not a difficult task. For three out of six datasets, we perform nearly as well if not better than state-of-the-art related work. On datasets where we are not the clear winner, we argued where the difference in classification score exists. Differences were found in the amount of features used where related work uses many more features and achieving slightly higher results, and in the preprocessing methods. It seems that preprocessing forms an important part of obtaining higher classification results than we have, but it is unsure what methods exactly fit this problem context, as the researchers do not describe the process well enough.

Moreover, our ablation study proved that trigrams provide a positive effect to the classification metric scores. In all four cases, trigrams achieved a similar or better results on the dataset as with the general or alternative feature set.

Finally, by inspecting the feature importances for the general and trigrams feature sets, we found that the protocol is generally unimportant for a random forest. Instead, the length of packets in a netflow followed by the amount of packets in a netflow, duration of a netflow and the inter-packet arrival time split the data most effectively in benign or malicious class bins.

4.4.2. Packet-level RF

Table 4.14 provides the classification results on the packet-level dataset CIC-DDoS2019 over 5 folds. The results are very impressive, with almost no classification errors for malicious packets. For benign

	UNSW-NB15		CIDDS-001		CICIDS2017	
Feat. #	Feature	Score	Feature	Score	Feature	Score
1	sbytes	0.1876	mean_bytes	0.3367	Bwd Packet Length Mean	0.2013
2	smean	0.1547	Bytes	0.2782	Total Length of Bwd Packets	0.1568
3	udp	0.0985	Duration	0.1894	Total Length of Fwd Packets	0.1435
4	ct_state_ttl	0.0667	Packets	0.1731	Fwd Packet Length Mean	0.0985
5	sttl	0.0604	TCP	0.0134	Total Backward Packets	0.0713
6	dbytes	0.0600	UDP	0.0051	Fwd IAT Mean	0.0698
7	dmean	0.0569	ICMP	0.0042	proto_6	0.0640
8	dur	0.0554			Total Fwd Packets	0.0638
9	dinpkt	0.0536			Flow Duration	0.0604
10	sinpkt	0.0515			proto_17	0.0438

	CSE-CIC-IDS2018		CIC-DDoS2019		inSDN	
Feat. #	Feature	Score	Feature	Score	Feature	Score
1	Flow Duration	0.2241	Bwd Pkt Len Mean	0.2573	Fwd Pkt Len Mean	0.1783
2	Fwd IAT Mean	0.1585	TotLen Bwd Pkts	0.1730	Bwd Pkt Len Mean	0.1629
3	TotLen Fwd Pkts	0.1247	Fwd Pkt Len Mean	0.1708	TotLen Bwd Pkts	0.1288
4	Fwd Pkt Len Mean	0.1137	TotLen Fwd Pkts	0.1606	Tot Fwd Pkts	0.0958
5	Fwd IAT Std	0.0780	Tot Fwd Pkts	0.0881	Tot Bwd Pkts	0.0852
6	Tot Bwd Pkts	0.0624	Flow Duration	0.0514	TotLen Fwd Pts	0.0697
7	Bwd Pkt Len Mean	0.0505	Tot Bwd Pkts	0.0308	Flow Duration	0.0674
8	TotLen Bwd Pkts	0.0503	Fwd IAT Mean	0.0249	Fwd IAT Mean	0.0674
9	Tot Fwd Pkts	0.0499	Bwd IAT Mean	0.0164	Bwd IAT Mean	0.0518
10	Bwd IAT Mean	0.0372	proto_6	0.0129	proto_17	0.0482

Table 4.12: Random forest top 10 important general features per connection-level dataset. Label of each feature is reported along with their total reduction of the gini score averaged over all trees.

	CIDDS-001		**CICIDS2017**	
Feat. #	Feature	Score	Feature	Score
1	Bytes_mean	0.2089	Fwd Packet Length Min_std	0.1594
2	Packets_mean	0.1925	Bwd Packet Length Min_std	0.1307
3	mean_bytes_mean	0.1654	Bwd Packet Length Max_mean	0.1113
4	Duration_mean	0.1132	Fwd Packet Length Max_std	0.0564
5	mean_bytes_std	0.1015	Total Length of Bwd Packets_std	0.0528
6	Bytes_std	0.1012	Total Backward Packets_std	0.0526
7	Duration_std	0.0997	Bwd Packet Length Max_std	0.0427
8	Packets_std	0.0175	Total Length of Bwd Packets_mean	0.0382
9			Fwd Packet Length Min_mean	0.0377
10			Flow IAT Max_mean	0.0376

	CIC-DDoS2019		**inSDN**	
Feat. #	Feature	Score	Feature	Score
1	Bwd Packet Length Max_mean	0.1735	Bwd Pkt Len Min_std	0.1352
2	Total Backward Packets_std	0.1155	Bwd Pkt Len Min_mean	0.1170
3	Bwd Packet Length Max_std	0.0982	Fwd Pkt Len Min_mean	0.1028
4	Total Length of Bwd Packets_mean	0.0874	Fwd Pkt Len Min_std	0.0960
5	Fwd Packet Length Min_mean	0.0777	Tot Bwd Pkts_mean	0.0837
6	Bwd Packet Length Min_mean	0.0612	TotLen Bwd Pkts_std	0.0686
7	Total Length of Bwd Packets_std	0.0585	TotLen Bwd Pkts_mean	0.0466
8	Total Length of Fwd Packets_mean	0.0562	Tot Fwd Pkts_mean	0.0415
9	Bwd Packet Length Min_std	0.0445	Tot Fwd Pkts_std	0.0352
10	Total Fwd Packets_mean	0.0391	Fwd Pkt Len Max_std	0.0351

Table 4.13: Random forest top 10 important trigram features per dataset. Label of each feature is reported along with their total reduction of the gini score averaged over all trees.

packets, the results are great as well, with a true negative rate of 99.73%. A high result was somewhat expected due to the repetitive attack vector in this dataset, but the sampling method of only selecting the testing day for classification most likely affected it as well. Especially for packet-level data, it is trivial to classify (D)DoS attacks as the malicious packets are all very similar, if not identical, and in high quantity. Nonetheless, it scores higher than the random forest implementation on the connection-level data. An explanation for the higher score may be found by inspecting the feature importances of the model.

	CIC-DDoS2019			
Iter.	TP	TN	FP	FN
1	7010425	144812	154	0
2	7010425	144554	412	0
3	7010421	144376	590	4
4	7010426	144781	184	0
5	7009842	144353	612	584
Mean	7010307.8	144575.2	390.4	117.6
σ	$\pm 0.003\%$	$\pm 0.13\%$	$\pm 198.30\%$	$\pm 49.66\%$

Table 4.14: Binary contingency tables for the CIC-DDoS2019 dataset using the Random Forest implementation on packet-level data.

The feature importances of the packet-level data are shown in Table 4.15. The findings here confirm the findings of our data exploration in Section 4.3.2. Comparing the feature set from the connection-level dataset with the packet-level dataset, the biggest difference is the presence of Frame TTL in the packet-level dataset. This feature might explain the higher classification score of the packet-level model, as opposed to the connection-level model. The ICMP protocol provides a relatively higher reduction of the gini score than anticipated, along with the IGMP protocol. Moreover, one of the attack vectors in the DDoS dataset is a SYN flood attack; it is surprising that the SYN flag (or host count of SYN occurrences) was not considered beneficial in the resulting model. However, this is explained primarily by the frame length feature. Out of 17.3 million SYN-flood marked packets, 17.268 million have a length of 40 bytes. We can separate those packets from other packets even better by including TTL values. 12.5 million SYN-flood packets have the TTL value of 243, followed by 4.7 million with the value 64. Thus, there is no need for a separate feature that tracks the usage of the SYN flag.

	CIC-DDoS2019	
Feat. #	Feature	Score
1	Frame length	0.4114
2	Frame TTL	0.1957
3	Proto_ICMP	0.1587
4	Proto_IGMP	0.1164
5	Proto_TCP	0.0344
6	Proto_UDP	0.0246
7	TCP_NS	0.0221
8	TCP_CWR	0.0148
9	TCP_ECE	0.0071
10	TCP_URG	0.0048

Table 4.15: Random forest top 10 important features for the packet-level dataset. Label of each feature is reported along with their total reduction of the gini score averaged over all trees.

Packet-level conclusions

In the packet-level experiment on CIC-DDoS2019 we have shown how important the frame length and time to live features are for packet-level classification. Especially for DDoS attacks, large amount of repetitive traffic is present and thus can be easily identified by looking at repetitive header values, although we did sample the train and test data from the same capturing day. Moreover, the protocol influences the type of attack that can take place, therefore attributing to the classification score as well. Even for TCP-based attacks, we found that TCP flags generally did not provide a good separation between malicious and benign traffic, apart from the NS flag in some cases. This is because most traffic can be explained based on protocol-agnostic features, i.e. frame length and time to live values.

4.5. Conclusions

With the results in mind, we can now answer our research question through our subquestions. The general classification model is successful in separating malicious from benign traffic on most connection-level datasets. With its worst true positive rate at 87.9% on NB15, our model generally competes on the level of other state-of-the-art models and on the inSDN dataset even heavily outperforms a LSTM+SVM model.

For the packet-level data, we do not have related work to compare to that classifies on packet data, but the results even outperform our connection-level feature set, on par with the state-of-the-art model by Lucky et al. [19].

For nearly all cases, our feature sets are also smaller than other state-of-the-art models, improving the simplicity of the model, yet retaining similar classification scores. Only for UNSW-NB15 the feature set is much larger than what is reported by related work. This is, as discussed in Section 4.3.1, due to the large list of one-hot encoded protocol values. In practice, almost none of the values contribute to the decision making process of the random forest model, but we chose not to further filter the feature list for simplicity.

Thus, to answer our first subquestion:

- It is indeed possible to create a general white-box classification model that works well on multiple network intrusion datasets.

Another finding we proved was the contribution of using trigrams features as opposed to single-flow features:

- Using features based on trigrams instead of single-flow features provides a classification score increase for nearly all connection-level datasets.

Even on the connection data level, using trigrams almost always works better than standard flow features, proving the usefulness of working with the sequential property of network traffic. In the end, three out of four possible datasets utilized trigrams and the one that did not - CIDDS-001 - had near identical results, but switched the classification results of the TPR and TNR.

The related work models that (slightly) outperformed ours are often do so because of a much larger feature set, possibly in combination with features that identify specific connections or machines. In other cases, related work showed that properly sampling the data to fix the class imbalance of network intrusion datasets positively affected the classification score. However, their sampling techniques are described insufficiently and their results cannot be reproduced, leaving us with no detailed insights on how to improve. Regardless, our generalized out-of-the-box random forest model competes with or outperforms the feature sets of related work while not being optimized specifically for a single dataset. Moreover, our model retains the insights of the decision-making process due to the white-box nature of random forests. Thus, we answer our second subquestion:

- Related work produce models that are often indeed more complex than necessary for the selected datasets.

With these conclusions, we can return to our main research question, if there is a relationship between the design quality of a dataset and the ability to achieve high classification scores on it. Primarily, datasets with a very limited attack vector are trivial to classify well on. CIC-DDoS2019 only contains DDoS attacks, CIDDS-001's set of attacks is limited as well and CICIDS2017 only has 8 general attack types as well. Conversely, the older UNSW-NB15 dataset was regarded as a good dataset at the time and still holds up in terms of achieving high classification results. Moreover, the CSE-CIC-IDS2018 dataset definitely is an update to its 2017 version, with an upgraded attack vector set, more diverse benign traffic and a much larger network infrastructure. Interestingly, the authors of the new inSDN dataset performed an in-depth analysis on the relevant attack vectors for SDN networks and provided a dataset with diverse attacks, yet our general classification model works very well on it. Most likely, the benign traffic patterns are very simple and easily distinguishable from malicious traffic. This is corroborated by a specificity of 99.8%, which is higher than the recall of 98.4%. The general property here refers to the relevant threat/benign landscape of a dataset, as concluded in research question 1. However, other dataset design properties do not seem to influence the ability to classify well on the respective datasets, therefore the conclusion is not definite.

Our base assumption, as defined in Chapter 1.2, states that realistic traffic data is more difficult to classify well on than datasets on which we can achieve near perfect classification results. Our classification results using a general classification model show that at least four out of six datasets are easy to classify well on and thus unrealistic. The other two datasets, UNSW-NB15 and CSE-CIC-IDS2018, are more realistic but also have their issues. UNSW-NB15 is outdated due to its traffic patterns, whereas CSE-CIC-IDS2018 still suffers from the inherent problems of its predecessor CICIDS2017. For instance, its enormous network infrastructure size generates approximately 220GB of network traffic, much of which is unneeded repetitive benign traffic that bloat the dataset. Moreover, there are still some data records that contain invalid values and are therefore unusable in their current state. Regardless, when compared to its predecessor CICIDS2017 and newer DDoS-variant CIC-DDoS2019, its network infrastructure, more modern attacks and up-to-date agent profiles result in a more difficult classification problem and is, therefore, more realistic and useful than the other datasets. The results achieved by Kilincer et al. do show a very high classification score, but it should be noted that it uses 80 features, including port information on a support vector machine. Their model would most likely not be viable in a real setting due to the amount of information it requires to process. One interesting note is that the network infrastructure for CIDDS-001 allowed external access, meaning that external agents could attack the external server of the dataset. Even though it contains real attack data, the dataset was still very easy to classify well on. As discussed before, we contribute this to the very limited attack variety the dataset had. Based on our results, we say we demonstrated the assumption in that more complex datasets, in terms of classification difficulty, are more realistic than the simpler datasets on which we easily achieve high classification scores.

We have shown that a common feature set of statistical features works well on all datasets, and for three out of six datasets even achieves state-of-the-art classification scores. Moreover, respecting the sequential property of network traffic by use of trigrams on connection-level data leads to an increase in classification score in nearly all cases. Finally, the effectiveness of white-box models has been illustrated here. The results confirm the unnecessary use of black-box models for classification on the used datasets. It is, therefore, better to use white-box models that provide insight in the decision making process. Finally, we demonstrated our assumption that realistic datasets are more difficult to classify well on, and subsequently conclude that most publicly available intrusion detection datasets are not (yet) realistic.

5 Limitations and Future Work

This Chapter concerns the limitations of this work and interesting future directions to explore.

5.1. Quality measure

For this thesis, we have selected six synthetic, publicly available network intrusion detection datasets, all of which had compatible connection-level data and one also a compatible packet-level format. Although the datasets used different traffic generation tools or different parameters within those tools, they remain synthetic and an approximation of real data. Since our goal is to judge the quality of these datasets, it would be very interesting to compare our findings to a dataset consisting of real data to further test our assumption that real datasets are more complex and thus more difficult to classify well on. However, datasets containing real data are often unlabelled, out-of-date or simply unavailable to the public domain.

For future work, an extension to the quality measure where a comparison is made to real data, both on basis of design metrics and classification results, would provide a more complete overview of what constitutes as real data and how different the synthetic datasets are. Moreover, this could be paired with a more in-depth analysis on how to design a high-quality dataset.

5.2. Feature sets

Our feature sets focussed on statistical features as a result of aggregating packet-level information to its summarised flow version. For the tested datasets, these proved to be very effective, even when compared to related work. However, Section 4.2.2 also described some papers using sliding windows and focussing more on temporal features in general. We included an ablation study regarding the effect of trigrams on connection-level data classification, but it would be interesting to see how it works on packet-level data. By grouping packets per connection and calculating new features based on the mean, standard deviation and other statistical measures, one could apply a sliding window of n packets per connection and compute new features per packet. Our only packet-level dataset could not be grouped well per connection. Only about 50% of packets belonged to a connection longer than one packet.

5.3. Classification methods

On both data levels, we used a random forest implementation. Although its results were great, it does not classify well for minority classes. The information criterion works by maximizing the amount of correctly classified instances, but it does not inherently assign weights to classes that contain far less instances than other classes. Each dataset has multiple attacks present, some more than others. For instance, more advanced attacks such as an infiltration attack occur far less frequently in the data than a bruteforce attack. The result is that the random forest prefers correctly classifying the bruteforce instances over the less prevalent infiltration instances.

There are two propositions for future work. First, undersampling prevalent and repetitive attacks will shift the focus of the model more towards other classes. Oversampling minority classes might work as well, although data exploration is very important here to confirm the effectiveness of oversampling, as more advanced attacks might not necessarily have enough dataset instances that occur closely together on some plane. Second, the need of classifying 100% of DDoS attack packets or flows is far lower than classifying a high amount of infiltration attacks, as a DDoS attack contains far more traffic than an infiltration attack by an advanced persistent threat, and could, therefore, be detected with far less packets if classification is done on the more general host-level. Assigning different classification weights to different attack vectors could be an interesting topic to steer the classification model towards more dangerous threats.

5.4. Usability of academic insights in business

Academic work in the field of (network) intrusion detection gives the impression that there is a lot of usable knowledge available for designing machine learning algorithms tasked with intrusion detection, yet we do not know if it actually is being used and possible reasons behind the decision. Analysing pcap traces of organisations still often works by manually inspecting the file, using set filters to possibly find intrusions. Machine learning solutions are very prevalent and successful in research, yet there is no wide-spread use of them in business scenarios, apart from some proprietary black-box solutions using deep learning. An analysis paired with interviews with SOC operators/analysts would provide valuable information as to why most work is still done manually.

6 Conclusion

In this work, we measured the quality of publicly available IDS datasets by performing a literature study on the design process of a dataset (RQ1), and by creating a general classification model that works on multiple selected datasets, to see if the findings from RQ1 influence the difficulty of classifying on a dataset (RQ2).

We found that the design process of datasets was often lacking in several fields, but most prominently in analysing the domain/usecase of the dataset and corresponding threat/benign traffic landscape. Some datasets suffered from a generalized or uninteresting domain, such as a SOHO (Small Office / Home Office) environment while intending to replicate a more complex environment. In nearly all cases, the benign traffic events were too simple. Simple internet browsing (HTTP(S)) and e-mailing were always included, yet there exists a far greater variety of benign behaviour within most domains. Moreover, good datasets still remain snapshots in time, and become obsolete after some time. To solve this, we provide direct improvements that can be made when designing a dataset. A more detailed focus on the type of network in terms of infrastructure and size would result in a more well-defined domain that also contributes to data provenance. Concerning traffic landscape, An ideal threat landscape is difficult to build, as attack vectors carry many variations on the same attack type. For this, we advise to consult recent threat or incident reports for the relevant industry. Some of the reports come with technical information that can be leveraged towards building more complex and relevant attack vectors. For benign traffic events, we urge dataset creators to include more variation. The datasets by the University of New Brunswick (UNB IDS) are a step in the right direction, by also including FTP and SSH traffic. However, the HTTP(S) traffic is still rather limited as it only encapsulates simple web browsing, yet it is much more diverse than that. For instance, video conferences or online meetings occur very frequently nowadays. Including those will make the benign traffic landscape more complex and realistic. Finally, to combat the ageing of datasets, the creation of them needs to be generalized towards a framework. One important step to do this has been explained by Shiravi et al. [34], by introducing the notion of agent profiles that each can simulate a different set and distribution of events. The University of New Brunswick implemented this in their creation process and now provide multiple up-to-date intrusion detection datasets. We recommend dataset creators to use this as well, such that they can update their datasets when needed and needing less time to do so.

The general classification model achieves good classification scores, for three datasets even similar to or better than state-of-the-art models made specifically for a single dataset. From our feature set analysis we found that respecting the sequential property of network traffic works well for classification, as our ablation study for trigrams showed an increase in classification score in nearly all cases. We further conclude that the solutions by most related work, especially the black-box solutions, are overly complex and do not provide enough classification score increase to offset the loss of insight. The complexity of the attack vectors in a dataset also greatly influence the difficulty of separating malicious and benign traffic. Attacks present in CIDDS-001 and CICIDS2017 contain either little variety or are not complex. As a result, they are easy to classify well on. The researchers of inSDN spent a lot of effort into building a representative and extensive attack set, yet we score very high there as well due to the simplicity of the benign traffic patterns. Other factors that influenced this were not found, however. The answer to our second research question remains, therefore, unsure.

Nevertheless, our assumption that realistic datasets are more complex and thus more difficult to classify well on holds. Datasets that suffer from a simple malicious or benign traffic composition are relatively easy to classify well on, and the update from CICIDS2017 to CSE-CIC-IDS2018 is a step in the right direction towards more realistic datasets.

In general, we can conclude that the available datasets do not (yet) represent realistic traffic and that the design methodology needs to be upgraded, as some datasets aim to be something they fail to make true. Once that happens, and more thought is put into analysing current traffic streams, more complex datasets could appear that meet the properties of a real dataset.

Bibliography

[1] Razan Abdulhammed, Miad Faezipour, Abdelshakour Abuzneid, and Arafat AbuMallouh. Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic. *IEEE sensors letters*, 3(1):1–4, 2018.

[2] Ahmet Aksoy, Sushil Louis, and Mehmet Hadi Gunes. Operating system fingerprinting via automated network traffic analysis. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 2502–2509. IEEE, 2017.

[3] Hayretdin Bahşi, Sven Nõmm, and Fabio Benedetto La Torre. Dimensionality reduction for machine learning based iot botnet detection. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1857–1862. IEEE, 2018.

[4] Davide Chicco. Ten quick tips for machine learning in computational biology. *BioData mining*, 10 (1):1–17, 2017.

[5] Naveed Chouhan, Asifullah Khan, et al. Network anomaly detection using channel boosted and residual learning based deep convolutional neural network. *Applied Soft Computing*, 83:105612, 2019.

[6] Gideon Creech and Jiankun Hu. Generation of a new ids test dataset: Time to retire the kdd collection. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 4487–4492. IEEE, 2013.

[7] Rohan Doshi, Noah Apthorpe, and Nick Feamster. Machine learning ddos detection for consumer internet of things devices. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 29–35. IEEE, 2018.

[8] Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. Ddosnet: A deep-learning model for detecting network attacks. In *2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 391–396. IEEE, 2020.

[9] Mahmoud Said Elsayed, Nhien-An Le-Khac, and Anca D Jurcut. Insdn: A novel sdn intrusion dataset. *IEEE Access*, 8:165263–165284, 2020.

[10] Osama Faker and Erdogan Dogdu. Intrusion detection using big data and deep learning techniques. In *Proceedings of the 2019 ACM Southeast Conference*, pages 86–93, 2019.

[11] Amirhossein Gharib, Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. An evaluation framework for intrusion detection dataset. In *2016 International Conference on Information Science and Security (ICISS)*, pages 1–6. IEEE, 2016.

[12] Waqas Haider, Jiankun Hu, Jill Slay, Benjamin P Turnbull, and Yi Xie. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. *Journal of Network and Computer Applications*, 87:185–192, 2017.

[13] Mohamed Idhammad, Karim Afdel, and Mustapha Belouch. Distributed intrusion detection system for cloud environments based on data mining techniques. *Procedia Computer Science*, 127:35–41, 2018.

[14] A. Kenyon, L. Deka, and D. Elizondo. Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets. *Computers & Security*, 99:102022, 2020. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2020.102022. URL http://www.sciencedirect.com/science/article/pii/S0167404820302959.

[15] Farrukh Aslam Khan, Abdu Gumaei, Abdelouahid Derhab, and Amir Hussain. A novel two-stage deep learning model for efficient network intrusion detection. *IEEE Access*, 7:30373–30385, 2019.

[16] Ilhan Firat Kilincer, Fatih Ertam, and Abdulkadir Sengur. Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Computer Networks*, 188:107840, 2021.

[17] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Stefanos Gritzalis. Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset. *IEEE Communications Surveys & Tutorials*, 18(1):184–208, 2016.

[18] XuKui Li, Wei Chen, Qianru Zhang, and Lifa Wu. Building auto-encoder intrusion detection system based on random forest feature selection. *Computers & Security*, page 101851, 2020.

[19] Godswill Lucky, Fred Jjunju, and Alan Marshall. A lightweight decision-tree algorithm for detecting ddos flooding attacks. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 382–389. IEEE, 2020.

[20] Marek Małowidzki, P Berezinski, and Michał Mazur. Network intrusion detection: Half a kingdom for a good dataset. In *Proceedings of NATO STO SAS-139 Workshop, Portugal*, 2015.

[21] John McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):262–294, 2000.

[22] S. Miller and C. Busby-Earle. The role of machine learning in botnet detection. In *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 359–364, 2016.

[23] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015.

[24] Quoc Phong Nguyen, Kar Wai Lim, Dinil Mon Divakaran, Kian Hsiang Low, and Mun Choon Chan. Gee: A gradient-based explainable variational autoencoder for network anomaly detection. In *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 91–99. IEEE, 2019.

[25] Hussein Oudah, Bogdan V Ghita, and Taimur Bakhshi. A novel features set for internet traffic classification using burstiness. In *ICISSP*, pages 397–404, 2019.

[26] Markus Ring, Sarah Wunderlich, Dominik Grüdl, Dieter Landes, and Andreas Hotho. Creation of flow-based data sets for intrusion detection. *Journal of Information Warfare*, 16:40–53, 2017.

[27] Markus Ring, Sarah Wunderlich, Dominik Grüdl, Dieter Landes, and Andreas Hotho. Flow-based benchmark data sets for intrusion detection. In *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS)*, pages 361–369. ACPI, 2017.

[28] Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, and Andreas Hotho. A survey of network-based intrusion detection data sets. *Computers & Security*, 86:147 – 167, 2019. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2019.06.005. URL http://www.sciencedirect.com/science/article/pii/S016740481930118X.

[29] Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. Network anomaly detection using lstm based autoencoder. In *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, pages 37–45, 2020.

[30] Vasileios Serentellos. Anomaly Detection in Network Traffic using Multivariate State Machines. Master's thesis, TU Delft, the Netherlands, 2020. URL http://resolver.tudelft.nl/uuid:003c0d5e-481a-4f52-8c3a-95c8dddcab9d.

[31] Iman Sharafaldin, Amirhossein Gharib, Arash Habibi Lashkari, and Ali Ghorbani. Towards a reliable intrusion detection benchmark dataset. *Software Networking*, 2017:177–200, 01 2017. doi: 10.13052/jsn2445-9739.2017.009.

[32] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP*, pages 108–116, 2018.

[33] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali A Ghorbani. Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In *2019 International Carnahan Conference on Security Technology (ICCST)*, pages 1–8. IEEE, 2019.

[34] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3): 357–374, 2012.

[35] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE, 2010.

[36] Jungsuk Song, Hiroki Takakura, Yasuo Okabe, Masashi Eto, Daisuke Inoue, and Koji Nakao. Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation. In *Proceedings of the first workshop on building analysis datasets and gathering experience returns for security*, pages 29–36, 2011.

[37] Michio Sugeno and Takahiro Yasukawa. A fuzzy-logic-based approach to qualitative modeling. *IEEE Transactions on fuzzy systems*, 1(1):7–31, 1993.

[38] D. S. Terzi, R. Terzi, and S. Sagiroglu. Big data analytics for network anomaly detection from netflow data. In *2017 International Conference on Computer Science and Engineering (UBMK)*, pages 592–597, 2017.

[39] Ankit Thakkar and Ritika Lohiya. A review of the advancement in intrusion detection datasets. *Procedia Computer Science*, 167:636–645, 2020.

[40] Vrizlynn LL Thing. Ieee 802.11 network anomaly detection and attack classification: A deep learning approach. In *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2017.

[41] Imtiaz Ullah and Qusay H Mahmoud. A two-level hybrid model for anomalous activity detection in iot networks. In *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2019.

[42] Yi Xie, Shensheng Tang, Yang Xiang, and Jiankun Hu. Resisting web proxy-based http attacks by temporal and spatial locality behavior. *IEEE transactions on parallel and distributed systems*, 24(7):1401–1410, 2012.

[43] Ying Zhong, Wenqi Chen, Zhiliang Wang, Yifan Chen, Kai Wang, Yahui Li, Xia Yin, Xingang Shi, Jiahai Yang, and Keqin Li. Helad: A novel network anomaly detection model based on heterogeneous ensemble learning. *Computer Networks*, 169:107049, 2020.