HERA: Enhancing Network Security with a New Dataset Creation Tool

Daniela Alexandra Pires Pinto
Master in Information Security
Department of Computer Science
Faculty of Sciences of the University of Porto
2024

HERA: Enhancing Network Security with a New Dataset Creation Tool

Daniela Alexandra Pires Pinto

Dissertation carried out as part of the Master's in Information Security

Department of Computer Science

2024

Supervisor
Doutora Eva Maia, Investigador Auxiliar,
Porto School of Engineering, Polytechnic of Porto (ISEP-IPP)

Co-supervisor
Doutora Ivone Amorim, Professor Auxiliar Convidado,
Faculty of Sciences of the University of Porto

Acknowledgements

I wish to express my sincere gratitude to everyone who permitted the completion of this thesis.

First and foremost, I would like to extend my heartfelt appreciation to my supervisors, Eva Maia and Ivone Amorim. Their constant feedback, guidance and expertise have profoundly shaped this research journey. Thank you for dedicating your time and effort to my work and for continually pushing me to embrace new challenges.

I also extend my thanks to Professor Isabel Praça and everyone at GECAD for their acceptance, guidance and support throughout the development of my work.

To my parents and brother André, words cannot fully express my gratitude for everything you have done for me. Your love and unconditional support have been my greatest privilege. I am who I am today because of you. Mom and Dad, thank you for giving me more than I could ever repay. André, you have always been my role model; thank you for your unwavering presence and for believing in me and my abilities. A special thanks to Lais, your kindness has not gone unnoticed. And to my grandmother, who never fails to wave at me while I work, your support means the world to me.

To my friends, Beatriz, Cláudia and Patrícia, who have known me since before I knew how the Internet worked, it has been a joy to grow up alongside you. Ema, thank you for being such a dear friend and for all our mood-lifting sessions. To all my other friends, thank you for your endless laughter and for making this journey so much more enjoyable.

Lastly, but certainly not least, to my dear Vasco. Over the past six years, you have been a cherished companion. Your unwavering belief in me every time I embark on a new journey has been crucial to my success. Your support and understanding are gifts for which I am eternally grateful.

To those mentioned above, and to all whose names may not appear here but who have contributed in one way or another, I extend my heartfelt thanks. And, of course, to my fur balls, Snoopy and Astolfinho, thank you for your calming cuteness; your presence was always a source of comfort.

Abstract

In recent years, the surge in cybersecurity threats has highlighted the critical need for robust network intrusion detection systems. These systems are essential for identifying anomalous behaviour within modern networks. By leveraging machine learning to process large amounts of data, network intrusion detection systems have become powerful tools for detecting patterns and predicting cybersecurity threats, thereby significantly enhancing network security. For continuous improvement of these systems and the machine learning models that aid it, access to high-quality network data is imperative. This data is typically provided through network intrusion detection datasets, which the research community actively develops. In this context, network traffic analysis tools, even though not initially designed for this purpose, are frequently used to create these datasets.

This thesis aims to develop an open-source tool to assist in the dataset creation process. For that a comprehensive review of existing tools, datasets, and features in the context of intrusion detection was carried out. A detailed study of the current traffic analysis tools was performed highlighting their advantages and drawbacks, and providing a categorized overview of them. Additionally, popular datasets from the research community are analysed, providing a timeline illustrating the advancements over the past twenty years. This analysis covers the evolution of dataset creation techniques, from packet-based to flow-based data, and the methods of traffic capture, whether real, simulated, or hybrid. Furthermore, this analysis examines how dataset features influence the effectiveness of network intrusion detection systems, providing a detailed review of commonly used features and their impact on detection accuracy.

Finally, in line with the main objective of the thesis, this work introduces a new open-source tool, HERA, that employs a *Flow Exporter* to aggregate flows from PCAP files and export feature sets tailored to the user's specific needs. Additionally, by using ground truth, the tool allows the labelling of the datasets. This tool simplifies the process of generating datasets from custom network scenarios, enhancing usability and efficiency for researchers. The tool was validated using PCAP files from the public datasets UNSW-NB15 and CIC-IDS2017.

Keywords: Intrusion Detection, Intrusion Detection Dataset, Network Traffic Analysis, Network Traffic Tools, Dataset Features

Resumo

Nos últimos anos, os aumentos de ataques informáticos têm salientado a necessidade para sistemas de deteção de intrusão em redes robustos. Estes sistemas são essenciais para identificar comportamentos anómalos em redes modernas. A aproveitar aprendizagem automática para processar grandes volumes de dados, estes sistemas tornaram-se ferramentas eficazes para detetar padrões e prever ameaças de cibersegurança, aumentando assim a segurança de redes. Para contínua melhoria destes sistemas e os modelos de aprendizagem automática que os auxiliam, o acesso a dados de rede de alta qualidade é imperativo. Estes dados são geralmente providenciados por *datasets* de deteção de intrusão de rede, que a comunidade académica tem ativamente desenvolvido. Neste contexto, ferramentas de análise de tráfego de rede, mesmo que não tenham inicialmente sido criadas para esse fim, são frequentemente usadas para auxiliar na criação de *datasets*.

Esta tese tem como objetivo desenvolver uma ferramenta de código aberto para auxiliar no processo de criação de *datasets*. Para isso, inclui uma revisão abrangente das ferramentas existentes, *datasets* e atributos no contexto de deteção de intrusão. Assim, foi realizado um estudo detalhado das atuais ferramentas de análise de tráfego, destacando as suas vantagens e desvantagens, fornecendo uma visão geral e categorizada das mesmas. Adicionalmente, *datasets* populares da comunidade académica são analisados, com uma linha temporal ilustrando os avanços ao longo dos últimos vinte anos. Esta análise abrange a evolução das técnicas utilizadas no processo de criação de *datasets*, no formato de pacotes ou *flows*, e os métodos de captura de tráfego, sejam reais, simulados ou híbridos. Além disso, a análise examina como atributos de *datasets* influênciam a eficácia dos sistemas de deteção de intrusão de redes, ao providenciar uma revisão detalhada de atributos comumente utilizados e o seu impacto na precisão de deteção.

Finalmente, em linha com o principal objetivo da tese, este trabalho apresenta uma nova ferramenta de código aberto, HERA, que utiliza um *Flow Exporter* para agregar *flows* a partir de ficheiros de captura de tráfego e exportar conjuntos de atributos adaptados às necessidades específicas do utilizador. Adicionalmente, ao utilizar dados de referência, a ferramenta permite a identificação do tipo de tráfego dos *datasets*. Esta ferramenta simplifica, assim, o processo de criação de *datasets* que usam cenários específicos, melhorando a usabilidade e eficácia para os investigadores. Para além disso, a ferramenta foi validada usando os ficheiros de captura de tráfego de *datasets* públicos, UNSW-NB15 e CIC-IDS2017.

Palavras-chave: Deteção de Intrusão, Dataset de Deteção de Intrusão, Análise de Tráfego de Rede, Ferramentas de Tráfego de Rede, Atributos de Datasets

Contents

List of Tables

List of Figures

Listings

Acronyms

ACM Association for Computing Machinery

AI Artificial Intelligence

AP Access Point

ARP Address Resolution Protocol

ATT&CK Adversarial Tactics, Techniques, and Common Knowledge

CERT Computer Emergency Response Team

CIC Canadian Institute for Cybersecurity

CLI Command Line Interface

CPU Central Processing Unit

CSV Comma-Separated Values

CSE Communications Security Establishment

D-ITG Distributed Internet Traffic Generator

DDoS Distributed Denial of Service

DNS Domain Name System

DoS Denial of Service

DOROTHEA **DO**cker-based f**R**amework f**O**r ga**TH**ering n**E**tflow tr**A**ffic

ENISA European Union Agency for Cybersecurity

FTP File Transfer Protocol

GMT Greenwich Mean Time

GUI Graphical User Interface

HERA **H**olistic n**E**twork featu**R**es **A**ggregator

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IETF Internet Engineering Task Force

ICMP Internet Control Message Protocol

ICS Industrial Control System

IEEE Institute of Electrical and Electronics Engineers

IDS Intrusion Detection System

IIoT Industrial Internet of Things

IP Internet Protocol

IPFIX IP Flow Information Export

IoT Internet of Things

ISP Internet Service Provider

LAN Local Area Network

MAC Medium Access Control

ML Machine Learning

MitM Man-in-the-Middle

NID Network Intrusion Detection

NIDS Network Intrusion Detection System

NTA Network Traffic Analysis

OISF Open Information Security Foundation

PCAP Packet Capture

PFCP Packet Forwarding Control Protocol

PRISMA Preferred Reporting Items for Systematic Reviews and Meta-Analyses

R2L Remote to Local Attack

RST TCP Reset

SDN Software Defined Networking

SSH Secure Shell

SMTP Simple Mail Transfer Protocol

SOHO Small Office/Home Office

SQL Structured Query Language

SiLK System for Internet-Level Knowledge

TCP Transmission Control Protocol

U2R User to Root Attack

UDP User Datagram Protocol

UTC Coordinated Universal Time

VLAN Virtual Local Area Network

WSN Wireless Sensor Networks

YAF Yet Another Flowmeter

Chapter 1

Introduction

This chapter provides a contextualization and detailed description of the research conducted in this thesis. It outlines the established objectives, the formulated research questions, and highlights the study's main contributions. Additionally, it concludes with an overview of the structure of the document.

1.1 Context and Motivation

Cybersecurity plays an increasingly crucial role in modern society, with Network Traffic Analysis (NTA) being a key component. This involves monitoring and analysing the data transmitted over a computer network to ensure network administrators can understand and optimize network performance [1]. This type of data is provided by NTA tools, designed to obtain traffic information to compile the necessary insights into the network. Despite not being initially designed for this purpose, the visibility that NTA tools offer into network activity and performance can be useful in detecting security threats.

On the other hand, Network Intrusion Detection Systems (NIDSs) were explicitly designed for security purposes, focusing on detecting malicious activities or unauthorized access attempts within network traffic [2]. NIDSs operates by continuously scanning network packets in real-time, searching for suspicious patterns or anomalies that may indicate a security breach. Upon detecting a potential threat, NIDSs can trigger alerts or take automated actions to mitigate the risk, such as blocking harmful traffic or alerting network administrators [3]. NIDSs can be categorized into two types: Signature-based and Anomaly-based systems [4]. Signature-based NIDSs identify threats using predefined patterns of known malicious activity (signatures). This makes them effective against known threats but limited against new attacks, making them vulnerable to zero-day attacks or polymorphic malware that is capable of changing their signature. Anomaly-based NIDSs detect deviations from normal network behaviour, being capable of identifying unknown threats but being prone to false positives. It is in this context that Artificial Intelligence (AI) provides significant enhancement by being incorporated into anomaly-based NIDSs helping

to analyse network traffic patterns and behaviours to identify potential threats [5]. For this reason, historical data is crucial to be leveraged by these systems. By learning from the data, they can adapt to emerging threats in real-time, which is a clear advantage over signature-based approaches. Furthermore, Machine Learning (ML) is capable of identifying subtle variations in network traffic, allowing AI to enhance the accuracy and reduce false positives in signature-based NIDSs [6, 7].

For the integration of AI, as mentioned, large amounts of data are required in the form of datasets. These can be used by researchers to identify common attack patterns, develop effective detection algorithms and evaluate the performance of NIDSs under various conditions [8]. Therefore, over the past two decades, numerous researchers have contributed to the development of Network Intrusion Detection (NID) datasets. The Canadian Institute for Cybersecurity (CIC)[1] has been particularly prominent in this effort, producing a leading family of datasets that are widely utilized in the field. However, recent shortcomings with labelling inaccuracies, data imbalances with over-representation of certain attacks and data duplication in these datasets, highlight the escalating need for high-quality datasets. Moreover, the problems associated with these particular datasets stem from issues within the tool used to generate the flows to populate the dataset, namely CICFlowMeter[2]. This underscores another critical requirement in dataset creation: the necessity for reliable tools to gather and analyse the data for NID datasets [9].

An additional critical aspect of dataset creation involves incorporating the appropriate data, organized in the form of features. The effectiveness of ML models is reliant on the quality and diversity of the datasets used, but also on the inclusion of appropriate features. These features are what allow ML models to classify network events and currently no standardization is being followed despite efforts to standardize a features set by the research community [10, 11]. This problem is particularly evident when considering that features are being pre-selected based on the researchers' domain knowledge. Consequently, it is necessary to perform feature selection after generating the datasets as they often contain numerous features that may not contribute to the efficacy of ML models.

This thesis explores these themes with an analysis of the NTA tools frequently used in the dataset creation process. Followed by a detailed analysis of the current datasets being used in NID research. Finally, a new tool, **H**olistic n**E**twork featu**R**es **A**ggregator (HERA), is provided to streamline the dataset creation process, leveraging a *Flow Exporter*. This tool allows for flow information to be used with different feature sets to choose from as an option. To conclude, the tool labels the dataset, providing all necessary information for a useful NID dataset.

[1] https://www.unb.ca/cic/

[2] https://www.unb.ca/cic/research/applications.html

1.2 Objectives and Research Questions

The **main objective** of this thesis was to construct an open-source tool to obtain data for traffic analysis. In alignment with this overarching goal, three more specific objectives were established:

- **Objective 01:** Investigate the state-of-the-art network analysis tools, the datasets created with data obtained by these and their corresponding features.
- **Objective 02:** Develop a tool to create datasets from Packet Capture (PCAP) files.
- **Objective 03:** Validate and test the implemented solution.

Furthermore, to guide the research and ensure the achievement of the defined objectives, a **main research question** was established: "What flow-based tools exist for network traffic analysis?". This central question led to the formulation of four additional research questions:

- **Research Question 01:** "What are the current state-of-the-art flow-based tools for network traffic analysis?"
- **Research Question 02:** "How do characteristics of network traffic analysis tools impact the generated NID datasets?"
- **Research Question 03:** "How do flow-level features influence network traffic analysis?"
- **Research Question 04:** "What are traffic analysis tools' prevailing limitations and challenges?"

1.3 Main Contributions

The research conducted in this thesis yielded significant findings and resulted in the development of a tool for producing NID datasets. Accordingly, the main contributions can be divided into two parts. The first part pertains to the contributions derived from the literature review and can be detailed as the following:

1. Review of the state of the art of existing NTA tools, assessing their capabilities and functionalities.
2. Grouping of NTA tools into categories based on their primary functions, facilitating a structured overview of available tools.
3. Presentation of a dataset creation workflow encompassing data collection or generation, flow creation, feature extraction, and labelling.

4. Review of the state of the art of existing NID datasets and creation of a timeline of their appearance.
5. Analysis of NID dataset features, including basic network information such as source and destination addresses and others, for example, packet statistics.
6. Categorization of dataset features into distinct groups, aiding in a systematic organization and understanding of the diverse range of features present in the different NID datasets.

This review has led to the preparation of a scientific publication, currently under review by Computer Networks:

- Pinto D., Amorim I., Maia E., and Praça I., "A Review on Intrusion Detection Datasets: Tools, Processes, and Features", *Computer Networks*.

The second part encompasses the contributions related to the development and testing phase, which consists of the following:

1. Implementation of HERA to generate flow-based datasets and traffic statistics using captured network traffic.
2. Testing and validation of HERA.
3. Anomaly detection using ML models and comparison of the dataset versions produced by HERA and the originals.

1.4 Document Structure

This document is structured into multiple chapters facilitating a clear and comprehensive understanding of the research conducted. They can be described as follows.

Chapter 1 outlines the motivations and context for this thesis. It presents the research questions and objectives that guide the investigation, as well as the main contributions of the work.

Chapter 2 presents a systematic literature review. This chapter is divided into an overview of the methodology employed and a subsequent exposition of the findings, categorized into three sections: tools, datasets, and features. The chapter concludes with a discussion of the findings for each category.

Chapter 3 describes the implementation process of the tool, including reviews on the requirements and underlying tools used in the final implementation. It concludes with an overview of the tool, demonstrating its workflow and elements with which the user interacts.

Chapter 4 provides the evaluation of the implemented tool with a focus on ML using two datasets, UNSW-NB15 and CIC-IDS2017, analysed during the state-of-the-art.

Chapter 5 describes the thesis's main conclusions, analysis of results, and achieved objectives, indicating the limitations of the project and possible future work.

Chapter 2

Related Work

This chapter presents the findings of the literature review based on the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) approach and the corresponding answers to the research questions posed in Section 1.2, namely the state-of-the-art Network Traffic Analysis (NTA) tools and datasets. Furthermore, the following sections delve into the evolution of these datasets used for network traffic analysis as well as a review of the features present in them.

2.1 Research Methodology

This section delineates the research methodology employed to perform a systematic review, adhering to the PRISMA framework [12]. PRISMA is an evidence-based standard for providing a structured and transparent format for literature reviews to authors, journal peer reviewers and editors, fostering reproducibility of the research results [13].

The PRISMA standard consists of a three-phase process: Identification, Screening and Inclusion. Each of these phases is described in detail in the subsequent sections. It is noteworthy that after the PRISMA standard was followed, a snowballing process of verifying sources led to the inclusion of more publications beyond the direct query result.

2.1.1 Identification Phase

The Identification Phase consists of a comprehensive search in electronic databases to identify relevant studies. For this work, the selected databases were the Association for Computing Machinery (ACM) Digital Library[3], Institute of Electrical and Electronics Engineers (IEEE) Xplore[4], Scopus[5] and Web of Science[6].

[3] https://dl.acm.org/
[4] https://ieeexplore.ieee.org/Xplore/home.jsp
[5] https://www.scopus.com/home.uri
[6] https://www.webofscience.com/wos/woscc/basic-search

Firstly, a query was defined to perform the research. This included the combination of different keywords alongside boolean operators (AND, OR) and the use of quotation marks to restrict the word used to an exact match. The utilization of quotation marks was selectively applied so that, when not in use, it facilitated the search of the plural versions of the term, thereby refining and broadening the scope of the query. The query was formulated to retrieve publications on network traffic, covering both attack and benign types. It focused on "flow" as the primary method of traffic analysis but also included publications that employed alternative methods. Additionally, terms associated with the generation of datasets for intrusion detection and security purposes were included, reflecting their widespread use for such endeavours.

Within these databases, the query presented in Listing 2.1 was employed. This query, individually modified to meet the specific querying criteria of each database, aimed to retrieve papers featuring these terms in their title, abstract and/or keywords.

```
( "network traffic" OR "network data" OR "network packets" ) AND ( "
    flow-based" OR "traffic flow" OR "network flow" OR "attack traffic" OR "
    benign traffic" OR NetFlow ) AND dataset AND ( "intrusion detection" OR
    "security" )
```

Listing 2.1: Query utilized in Identification Phase.

In total, the query performed on October 31, 2023, yielded 859 results across the different databases. Among these results, 368 results were duplicated, resulting in 491 unique publications progressing on to the next phase, the Screening Phase. Table 2.1 provides a breakdown of the obtained results from each database.

Table 2.1: Number of results of the query by database.

Database	**Result**
ACM Digital Library	15
IEEE Xplore	186
Scopus	444
Web of Science	214
Total	859

A visual representation of the PRISMA process can be observed in Figure 2.1.

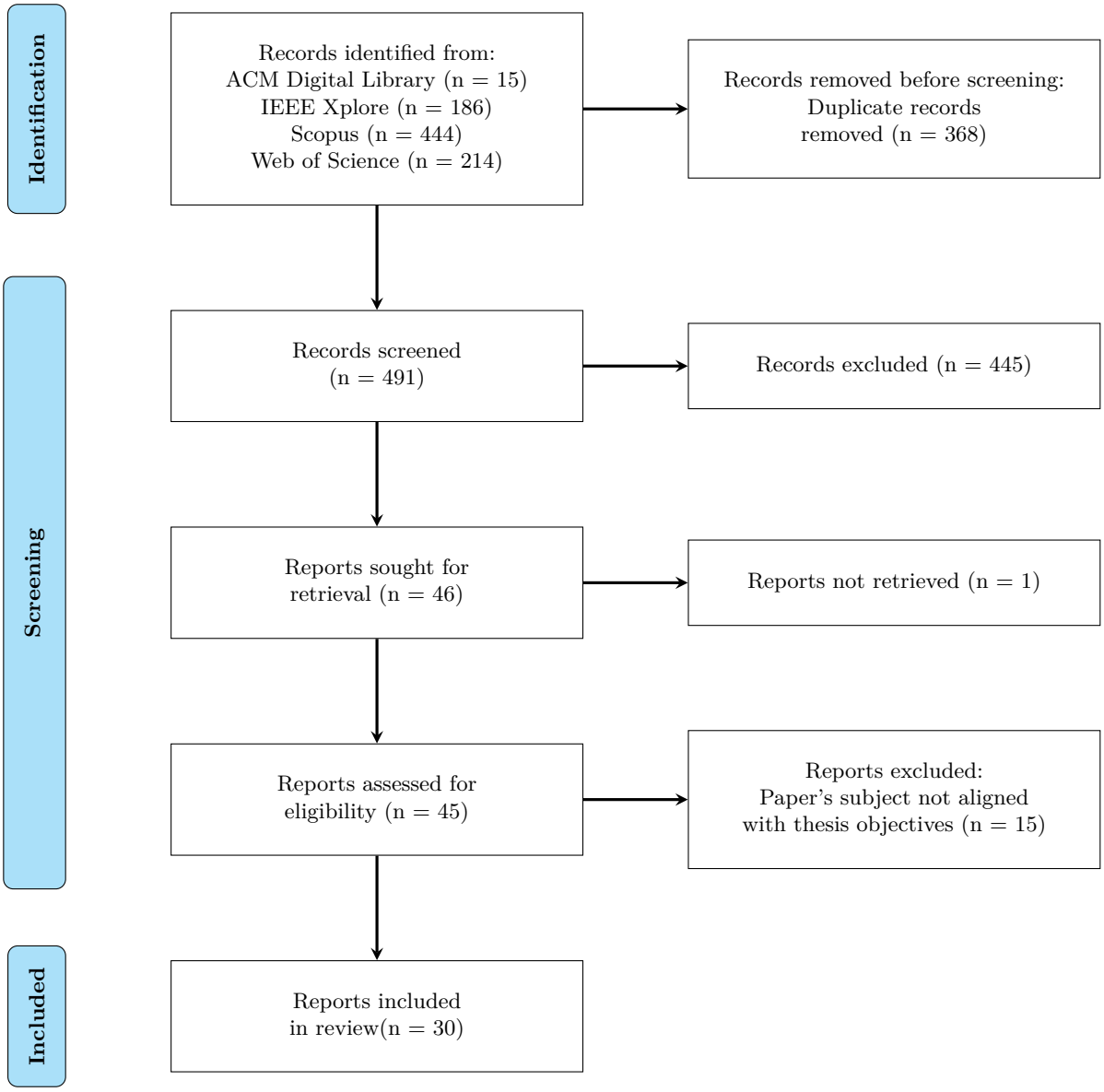


Figure 2.1: PRISMA Flowchart

2.1.2 Screening Phase

The Screening Phase consisted of a three-step procedure. Initially, papers underwent screening based on the examination of their title and abstracts. Subsequently, verification was conducted to ascertain the accessibility of each paper for further reading. Finally, a review of the entire publication was undertaken to assess their eligibility for inclusion in the systematic review.

2.1.2.1 Screening

The screening process began with the 491 publications carried over from the previous phase. During the evaluation of the title and abstracts, specific inclusion and exclusion criteria were considered that are summarized in Table 2.2. For inclusion criteria, it was considered publications that had relevance to traffic analysis, such as focusing on details of the creation, comparison or analysis of datasets and/or analysis tools, or papers that also performed a study on flow-level features. Furthermore, to include publications, they had to be written in English. The exclusion criteria for this thesis consisted of publications that did not specifically relate to tools for network traffic analysis, datasets and dataset features. Additionally, publications that only discussed the use of tools and datasets for developing algorithms and methodologies using Machine Learning (ML) to classify different types of attacks were also excluded. Following this screening process, 445 publications were excluded, resulting in 46 papers deemed eligible for further evaluation in the following steps.

Table 2.2: Inclusion and Exclusion Criteria in the Screening Phase

Inclusion Criteria	Exclusion Criteria
Relevant to traffic analysis.	Topics outside the scope of the research questions (eg. ML, neural network, cloud computing).
Specifies NID datasets.	Focus on the use of tools and datasets for ML algorithms.
Specifies traffic analysis tools.	Focus exclusively on cyber attacks.
Performed a study on flow-based features.	
Available in the English language.	

2.1.2.2 Retrieval

The retrieval process consists of preparing the articles for full-text screening, for this, a search for the full text is conducted and papers where retrieval is not possible are removed. One paper was identified as inaccessible from the 46 previously obtained, resulting in 45 papers available for a thorough evaluation of their eligibility.

2.1.2.3 Eligibility

For the final phase of the screening process where eligibility was assessed, a reading of the entire publication allowed to determine that 15 publications failed to meet the criteria for inclusion presented in Table 2.2. The papers were excluded because they did not primarily focus on the analysis of flow-based features or the datasets and analysis tools. Consequently, this concluded the screening phase with 30 papers proceeding to the final phase.

2.1.3 Inclusion Phase

Upon the completion of the screening phase, a total of 30 papers were considered eligible for in-depth examination. From this analysis, using a snowballing approach, 36 additional studies were identified that focused on creating new datasets or tools that can be used to create datasets. This process uncovered datasets from the last two decades that the PRISMA process was unable to find, as well as lesser-known tools. To conclude, a total of 66 papers were considered eligible to extract information and develop an understanding of the current state-of-the-art.

2.2 Categorization of Network Traffic Analysis Tools

In the context of traffic analysis, various functionalities are necessary to fulfil specific tasks. For example, creating a dataset requires specific functionalities, namely collecting, extracting, and analysing traffic data. At times, these functionalities may all be consolidated in a singular tool or may be performed by a collection of distinct tools. This section explores the various NTA tools currently in use, identified during the systematic review, and their different classifications, focusing on their role in the generation of datasets.

Before proceeding to the classification of the tools reviewed, it is necessary to acknowledge that traffic analysis can be conducted through the analysis of individual packets previously collected or of flows. In this thesis, flows will be the focus of both the review and the work performed.

A flow is defined as the unidirectional sequence of Internet Protocol (IP) packets sharing identical values for the following attributes: source and destination of the traffic, routing information, and the protocol employed, as Cisco[7] specifies with version 5 of NetFlow [14]. It is, however, noteworthy that most of the presented tools generate bidirectional flows, or offer an option to use uni or bidirectionality. Additionally, there is another version of the protocol NetFlow that can be used, version 9, with other versions being obsolete. Although version 5, which has a fixed format, is the most commonly used version, it does not support IPv6 traffic, Medium Access Control (MAC) addresses, and Virtual Local Area Networks (VLANs), among other fields, which version 9, with a template-based format, permits [15].

Besides this protocol, other versions exist that function similarly to NetFlow that are also proprietary, such as jFlow which presents equivalent standards but was developed by Juniper Networks[8]. Others, allow for multi-vendor environments, such as sFlow by InMon Corporation[9] which uses statistical sampling and IP Flow Information Export (IPFIX) created by Internet Engineering Task Force (IETF)[10] which is derived from NetFlow version 9 but more flexible [16].

Moreover, the results of the state-of-the-art allowed the conclusion that NTA tools could be classified into six distinct categories, namely, *Flow Exporter*, with or without Feature Extraction, *Flow Collector*, *Generator*, *Analyser*, *Capturer* and *Classifier*. It is important to notice that certain tools may fall under one or more categories because, beyond their primary function, they can have other capabilities. The subsequent sections present a description of each type of classification and the current tools studied within each distinct category.

[7] https://www.cisco.com/c/pt_pt/index.html
[8] https://www.juniper.net/
[9] https://inmon.com/
[10] https://www.ietf.org/

2.2.1 Flow Exporter

Flow Exporters aggregate packets into flows and subsequently export the generated flow records into one or more *Flow Collectors*. To accomplish this, these tools can read the packets from captured network files or from a network interface. Afterwards, the *Flow Exporter* decodes the packets into flows by grouping packet properties into a set of values defined by the tool. This strategy results in the loss of granularity as analysing at the packet level offers a full overview of the network. However, the flow-level analysis offers advantages over packet analysis, as it requires less storage and processing power while offering a higher-level view of the network [16].

Certain *Flow Exporters* may have the capability of feature extraction within this context, while others may not. Sometimes it is also the case that the primary function of a tool is feature extraction, even if they are essentially a *Flow Exporter*. Feature extraction is the process by which raw data gets turned into features that contain relevant information regarding the data they represent. In the case of this thesis, features allow us to better understand and analyse flows. These features might be packet count, flow duration, protocol types or flow direction, among many others. They allow the identification of patterns within a network, making flow-level analysis more efficient.

Among the tools discovered in this review, those identified with *Flow Exporter* capabilities include CICFlowMeter[11], LycoSTand[12], nProbe[13], Softflowd[14], Yet Another Flowmeter (YAF)[15], Argus[16], ICSFlowGenerator[17] and other tools not publicly available, LiteEx [17], CSTITool [18] and HDFEF [19] in Section 2.2.1.8. Besides, there are other tools with this capability but not having it as a main function, namely nfdump, NFStream and Tranalyzer, in Sections 2.2.2.1, 2.2.4.1 and 2.2.4.3 respectively. The following sections describe these tools.

2.2.1.1 CICFlowMeter

CICFlowMeter[11], also formerly known as ISCXFlowMeter, was developed by the Canadian Institute for Cybersecurity (CIC)[18]. Habibi Lashkari et al. [20, 21] categorize the tool as a flow *Generator*, notwithstanding its functionality is that of a *Flow Exporter* which aggregates flows bi-directionally from packets, where the first packet determines the forward (source to destination), and backwards (destination to source). The tool yields over 80 distinct traffic features, providing flexibility to select from the existing features or to add new ones, where the output generated is a Comma-Separated Values (CSV) file. Additionally, the tool is capable of capturing the network packets or reading them from a file, converting them into flows. Furthermore, it provides the ability to analyse the flows created.

[11] https://www.unb.ca/cic/research/applications.html
[12] https://lycos-ids.univ-lemans.fr/
[13] https://www.ntop.org/products/NetFlow/nprobe/
[14] https://github.com/irino/softflowd
[15] https://tools.netsa.cert.org/yaf/
[16] https://openargus.org/
[17] https://github.com/AlirezaDehlaghi/ICSFlow
[18] https://www.unb.ca/cic/

Given the importance of CICFlowMeter, several researchers have analysed not only this tool, but also the datasets generated using it. For instance, in 2021, Engelen et al. [22] analysed the tool and the CIC-IDS2017 dataset, created by CIC using this tool, revealing errors in both. Regarding the tool, they identified issues concerning the Transmission Control Protocol (TCP) termination. Specifically, CICFlowMeter identified a connection as closed when it detected a FIN flag. However, according to the TCP specification, the FIN flag indicates only the termination of data transmission by the sender, and the connection itself is terminated only after both sides exchange this flag. Furthermore, regarding flow construction, a timing-related flaw was uncovered, wherein the closure of one flow can cause the subsequent flow to go in the wrong direction, resulting in the interchange of source and destination. Finally, the TCP Reset (RST) flag is used to reset a TCP flag, CICFlowMeter does not consider it as a way to terminate a TCP flow. Besides the criticism of the flow construction, the authors observed that the feature extraction process contains attributes that should be excluded during ML training. They also proposed a patch correcting the errors they found in the tool. It is noteworthy that after this publication, the tool has been updated and is now in version 4 on their GitHub page[19].

In 2022, Liu et al. [9] extended Engelen et al. [22]'s analysis by including the CSE-CIC-IDS2018 dataset. Their work presents further modifications to the CICFlowMeter tool, addressing the previously identified issues. Several incorrectly implemented functions to create features resulted in a mislabeling error during the generation of the dataset. As a result, features that would provide flow disambiguation were not presented accurately. The modifications introduced a patch to resolve issues related to packet timestamping, TCP segmentation offset, and correction of calculations of existing attributes. Moreover, they introduced new functionalities, such as total TCP flow duration, support for the Internet Control Message Protocol (ICMP), forward and backwards RST flag counts, and Coordinated Universal Time (UTC) timestamp with microsecond precision.

In 2021 and 2022, Rosay et al. [23, 24] further expanded on issues they identified. Specifically within CICFlowMeter, they observed feature duplication, feature miscalculation and wrong protocol detection. In addition, errors akin to those reported by Engelen et al. [22] and Liu et al. [9] were noted in the TCP termination flows. Their proposed solution consisted of the creation of another tool, called LycoSTand, which is detailed in Section 2.2.1.2.

In 2023, Lanvin et al. [25] used the work by Engelen et al. [22] and discarded the work by Rosay et al. [23] because their solution with the TCP termination could potentially lead to incorrect statistics. Adding to the work of Engelen et al. [22], they found more issues where CICFlowMeter fails to correctly create flow descriptions, generates incoherent timestamps, duplicates data and omits an attack. Afterwards, the authors present a fixed version of CICFlowMeter to correct these problems.

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

2.2.1.2 LycoSTand

LycoSTand[20] was a tool created in response to issues the authors Rosay et al. [23, 24] encountered in CICFlowMeter. It generates the same features produced by CICFlowMeter but with the corrections to the problems detailed in Section 2.2.1.1. Functioning as a *Flow Exporter* and *Analyser* at flow level, LycoSTand forms flows and extracts features. It uses Packet Capture (PCAP) files as input and calculates over 80 flow-based features, generating a CSV file.

2.2.1.3 nProbe

The software tool nProbe[21], developed by the ntop[22] company, is accessible both as a standalone application and in an embedded system, nBox[23]. Designed to operate with both NetFlow or IPFIX protocols, nProbe serves dual roles as both an *Flow Exporter* and a *Flow Collector*, being primarily recognized for its exporting capabilities. Besides acting as a *Flow Collector*, it also allows sending network data to *Flow Collectors*, acting only as an *Flow Exporter*. nProbe exhibits capabilities which include the capture, collection, exportation of flows and extraction of features.

2.2.1.4 Softflowd

Softflowd [24] functions as a *Flow Exporter* with additional capabilities as a feature extractor and flow *Analyser*, compatible with NetFlow/IPFIX. The tool creates the flows by either reading a PCAP or by listening on a network interface, requiring the libpcap library for capturing functions. Libpcap[25] is an open-source C++ library to capture and filter packets being mainly used by tools with packet-sniffing capabilities producing PCAP files. While Softflowd does not operate as a *Flow Collector*, it can transmit aggregated flows to a *Flow Collector* or internally summarize them for analysis, thereby encompassing some *Flow Collector*-like functionalities. Notably, softflowd is specialised in tracking active traffic flows that become inactive after a certain period. This cumulative expired information gets turned into statistics viewable by softflowctl, a program of softflowd to print that information on the command line.

[20] https://lycos-ids.univ-lemans.fr/
[21] https://www.ntop.org/products/NetFlow/nprobe/
[22] https://www.ntop.org/
[23] https://www.ntop.org/nbox/hardware-appliances/nbox/
[24] https://github.com/irino/softflowd
[25] https://github.com/the-tcpdump-group/libpcap

2.2.1.5 YAF

YAF[26] serves as a *Flow Exporter* with some feature extraction, developed by the Computer Emergency Response Team (CERT) Division at the Software Engineering Institute of Carnegie Mellon University[27]. Initially conceived as an experiment for IPFIX developments for bidirectional traffic flow, this tool allows capturing traffic in real-time or from PCAPs, creating flows, and complying with the IPFIX standard. YAF enables further analysis using other tools specifically designed for it, such as SiLK (Section 2.2.2.2), which also belongs to the CERT NetSA Security Suite.

In 2017, Fernandez et al. [26] examined various resources, such as available memory, Central Processing Unit (CPU) load, and volume of traffic monitored by *Flow Exporters* and *Flow Collectors*. Focusing on *Flow Exporters*, the authors selected nProbe (Section 2.2.1.3), softflowd (Section 2.2.1.4), and YAF, observing minimal disparities in the performance of these tools concerning traffic volume and available memory. However, in terms of CPU load, softflowd outperformed the others as it is not complex, albeit at the expense of being less configurable.

The main distinction found was the significant difference in the number of exported packets and average packet size among the tools, despite using the same PCAP file. Specifically, YAF produced the fewest packets with the largest packet size, while softflowd generated the highest number of packets with the smallest average packet size.

2.2.1.6 Argus

Argus[28], that stands for Audit Record Generation and Usage System, constitutes an open-source project developed by Carter Bullard at Georgia Tech[29], initially conceived as a network flow system and subsequently adapted for incident response purposes. The framework comprises two distinct components: the argus-server functions as a network flow sensor responsible for capturing or processing the raw network packets and generating the corresponding flows, while the argus-client is tasked with exporting the resultant features. Argus is compatible with various flow data technologies, including but not limited to NetFlow, IPFIX and jFlow. In addition to its capabilities as a *Flow Exporter*, *Flow Collector* and packet *Capturer*, Argus provides flow-level analysis functionalities.

[26] https://tools.netsa.cert.org/yaf/
[27] https://www.sei.cmu.edu/about/divisions/cert/
[28] https://openargus.org/
[29] https://www.gatech.edu/

2.2.1.7 ICSFlowGenerator

ICSFlowGenerator[30], developed by Dehlaghi-Ghadim et al. [27], functions as a *Flow Exporter* with the capacity to extract a total of 54 flow features, including 4 of which are labels. The tool can acquire the packets to transform into flows by either reading from a PCAP file or listening on a network interface. Following the creation of the flows and extraction of the features, ICSFlowGenerator creates labels to identify whether a flow is malicious. It can also generate predicated labels, that analyse the flows and predict their anomalies, having as such a *Classifier* capability.

2.2.1.8 Other NTA Tools

In addition to the tools already mentioned, other tools in the eligible publications were found which, to the best of known knowledge, are not publicly accessible. Therefore, these tools are here described based on the authors' publications.

Swarnkar et al. [17] developed **LiteEx**, a feature extraction tool to be lightweight and flexible with a simple Graphical User Interface (GUI). This tool is capable of extracting 68 packet-level features and 11 flow-level features with a low use of RAM and CPU. It is also capable of extracting these features from pre-captured network traffic traces in PCAP format. To do so, and because flow construction is computationally heavy, LiteEX constructs the flow temporarily only to be able to extract the features. For this reason, it has a *Flow Exporter*-like behaviour, but only for a small portion of its workflow.

Hsupeng et al. [18] developed **CSTITool** based on CICFlowMeter. This tool was created to export features while focusing on improving model performance when applying ML methods for the classification of attacks. It is capable of extracting 53 additional features to those obtained with CICFlowMeter. These focus especially on adding more information regarding the network identifiers of the flow, as well as its descriptors and distribution of bytes. The authors also added flow data that they named "domain knowledge" as they were attributes that stored statistics defined by experts. Their tests showed an improvement in performance metrics when their features were extracted with CSTITool.

Li et al. [19] created **HDFEF**, a framework that is able to dynamically update itself according to both network and flow-level features. It can detect and classify network flows, as well as detect malicious ones. Besides extracting flow-level features, it also extracts packet-level features, but based on the network flows.

[30] https://github.com/AlirezaDehlaghi/ICSFlow

2.2.2 Flow Collector

Flow Collectors are responsible for the reception, storage and pre-processing of flow data from the *Flow Exporter*. These tools present the advantage of collecting data sent by many *Flow Exporters*, aggregating the traffic flows for easier analysis. The data storage can be done in a database or in files, depending on the volume of the network and the requirements of the user. This allows for bigger networks to be monitored and for their records to be saved into datasets as *Flow Collectors* can combine data while improving its quality by processing it, for example by removing duplicate flows. *Flow Collectors* are also especially useful to be used alongside other tools when monitoring a network [26].

Tools with *Flow Collector* functions include the previously mentioned nProbe, Softflowd and Argus, described in Sections 2.2.1.3, 2.2.1.4 and 2.2.1.6 respectively, as well as the nfdump tools[31] and SiLK[32], detailed below.

2.2.2.1 nfdump tools

The nfdump[31] collection is comprised of different processing tools, designed to support versions 5, 7 and 9 of the NetFlow protocol. Within this toolset, several noteworthy tools are included. For instance:

- nfcapd - responsible for collecting flows transmitted by *Flow Exporters* and storing them in files. It also has a variant specifically designed for sFlow, an alternative protocol standard to NetFlow that employs packet sampling. This variant is known as sfcapd.
- nfdump - tasked with processing the NetFlow data stored by nfcapd, this tool serves as a method to filter and query flow records for subsequent analysis.
- nfanon - responsible for the anonymization of NetFlow records.
- nfreplay - responsible for replaying NetFlow data previously stored by nfcapd, facilitating testing and simulation.
- nfpcapd - responsible for capturing network packets either from an interface or from a pre-recorded file, converting them into nfdump records.
- NfSen - serves as the graphical web-based front-end for the nfdump tools.

Primarily functioning as a *Flow Collector* and *Analyser*, the nfdump toolset has incorporated recent additions to include *Flow Exporter* and packet-capturing functionalities [28].

[31] https://github.com/phaag/nfdump
[32] https://tools.netsa.cert.org/silk/

2.2.2.2 System for Internet-Level Knowledge (SiLK)

SiLK[33], which was developed by CERT[34] is a suite of traffic analysis tools allowing for the collection, storage and analysis of network flow data. SiLK is often used in conjunction with YAF (Section 2.2.1.5), as this tool will generate the records and SiLK provides the tools for processing them and analysing them.

In 2017, in the same paper referred to in Section 2.2.1.5, Fernandez et al. [26] also examined the resources used by three tools with capabilities of *Flow Collectors*, nProbe (Section 2.2.1.3), SiLK and the nfdump tool, nfcapd (Section 2.2.2.1). The authors found that the tools behaved similarly in terms of CPU load and memory consumption.

2.2.3 Generator

Traffic *Generators* simulate synthetic traffic at different levels, such as network, packet, or flow. They are useful for simulating traffic when it is not possible to create certain scenarios with real traffic-producing machines. Using this approach also allows for complete control over the produced traffic when trying to create datasets.

Among these *Generators* are the DOROTHEA[35], Harpoon[36], PackETH[37], Ostinato[38] and Distributed Internet Traffic Generator (D-ITG)[39] tools.

2.2.3.1 DOROTHEA

Campazas-Vega et al. [29] created **DO**cker-based f**R**amework f**O**r ga**TH**ering n**E**tflow tr**A**ffic (DOROTHEA) which is a framework created using Docker[40]. This tool presents a virtual solution as it simulates and generates traffic and collects flow data. It is capable of simulating both benign and malicious network traffic and easily labelling it. The framework allows for scalability and flexibility, enabling users to customize scripts for generating both benign and malicious traffic. Additionally, it facilitates the creation of network topologies, demonstrating simulation capabilities at network, packet, and flow levels.

[33] https://tools.netsa.cert.org/silk/
[34] https://www.sei.cmu.edu/about/divisions/cert/
[35] https://seguridad.unileon.es/index.php/DOROTHEA
[36] https://github.com/jsommers/harpoon
[37] https://github.com/jemcek/packETH
[38] https://github.com/pstavirs/ostinato
[39] https://github.com/jbucar/ditg
[40] https://www.docker.com/

2.2.3.2 Harpoon

Harpoon is a flow-level traffic *Generator* proposed by Sommers et al. [30], which is configurable and imitates flows by creating TCP and User Datagram Protocol (UDP) packets that have the same characteristics as the ones captured from live routers. It is a tool useful for producing background traffic for testing.

2.2.3.3 PackETH

PackETH is a simple packet *Generator* that permits the creation and sending of any packet or sequence of packets with the ability to adjust its parameters. The GUI version of the tool allows to create and send packets while the Command Line Interface (CLI) version only permits sending packets previously stored from a PCAP file.

2.2.3.4 Ostinato

Ostinato is a packet-level *Generator* capable of importing, editing, replaying and saving PCAP files. It permits packet crafting, allowing for setting a value for any field of any protocol with multiple streams at the same time, and with the possibility of generating statistics in real-time.

2.2.3.5 D-ITG

Botta et al. [31] created D-ITG[41], a tool with the capability to produce traffic at the packet level in the network, transport and application layers. D-ITG allows for user-defined traffic profiles that imitate real-world scenarios, such as changing traffic rates, packet sizes, protocols and others. It is also a tool that allows for the creation of different traffic scenarios by configuring hosts and their interactions.

2.2.4 Analyser

Analysers inspect, examine and analyse data at network, packet or flow level. These tools permit a deeper understanding of the network and allow administrators to change policies to improve and optimize network performance. This section expands on five *Analyser* tools, NFStream[42], BRO-IDS/Zeek[43], Tranalyzer[44], Snort[45] and Suricata[46].

[41] https://github.com/jbucar/ditg
[42] https://github.com/nfstream/nfstream
[43] https://zeek.org/
[44] https://tranalyzer.com/
[45] https://www.snort.org/
[46] https://suricata.io/

Other tools with some of these capabilities were the previously mentioned, such as CIC-FlowMeter, LycoSTand, Argus and nfdump, in Sections 2.2.1.1, 2.2.1.2, 2.2.1.6 and 2.2.2.1 respectively. However, two other popular capturing tools, Wireshark (Section 2.2.5.1) and Tcpdump (Section 2.2.5.2), also have this capability.

2.2.4.1 NFStream

Aouini and Pekar [32] created NFStream, a framework to analyse network data to facilitate ML. It can perform flow aggregation, making it a tool with the capability to flow export. However, it uses a seven-tuple, instead of the standard five-tuple, to define the flow incorporating two additional fields related to VLAN identifiers. This tool is also capable of extracting more than 80 features at the flow level. Using the information it collects, NFStream can also classify traffic flows. Besides this, by also using the libpcap[47] library, it is capable of traffic capture, making it the most well-rounded tool in the list presented.

2.2.4.2 BRO-IDS/Zeek

BRO-IDS/Zeek is a network analysis framework that can also be used as an Network Intrusion Detection System (NIDS). It was formerly known as BRO-IDS and was developed in 1995 by Lawrence Berkeley National Laboratory[48]. As an analysis framework, it produces logs that are used to monitor a network. Moreover, through the use of plugins and add-ons, BRO-IDS/Zeek can extract features from flows. It can also read PCAP files and capture live traffic.

2.2.4.3 Tranalyzer

Burschka and Dupasquier [33] created Tranalyzer[49], a tool working primarily as an *Analyser*, also capable of flow exporting by aggregating packets and feature extraction to help monitor intrusions. Tranalyzer takes already captured traffic as input or captures it from Ethernet interfaces. Furthermore, it supports a process of reducing PCAP files for analysis by selectively extracting and storing relevant information while discarding redundant data. Afterwards, during the aggregation of the flow, it does not create the standard five-tuple, but a six to nine tuple where it includes the VLAN, MAC addresses and EtherType, decreasing the number of flows and giving a higher level overview of the network. When testing Tranalyzer against tools such as BRO-IDS/Zeek and Wireshark, the authors found that their tool presented a better runtime.

[47] https://github.com/the-tcpdump-group/libpcap
[48] https://www.lbl.gov/
[49] https://tranalyzer.com/

2.2.4.4 Snort

Snort[50] is an open-source NIDS with *Analyser* capabilities developed by Cisco[51]. Snort can operate in three different modes. In Sniffer mode, the program acts as a packet sniffer, capturing network packets and displaying them on the console. In Packet Logger mode, it stores logs on disk. In NIDS mode, Snort monitors network traffic and analyses it based on predefined rules. In this mode, Snort is also capable of classifying traffic as either benign or malicious.

2.2.4.5 Suricata

Suricata[52] is another open-source NIDS developed by Open Information Security Foundation (OISF)[53]. It allows for packet analysis and capture of real-time traffic much like Snort, as well as the classification of traffic as either normal or potentially malicious.

2.2.5 Capturer

Capturers capture information on communication sessions between devices on a network and may save it to a file, usually, PCAP files. They allow the analysis of the network at a later time and the replayability of attacks if needed.

Previously mentioned tools with this capability include: CICFlowMeter, nProbe, Softflowd, YAF, Argus, ICSFlowGenerator, nfdump, NFStream, BRO-IDS/Zeek, Tranalyzer, Snort and Suricata described in the previous sections. Although these tools have this ability, mostly due to the use of the libcap library, the most well-known among them are Wireshark[54] and Tcpdump[55].

2.2.5.1 Wireshark (old Ethereal)

Wireshark[54], previously known as Ethereal, is a free and open-source packet *Analyser* with the ability to capture packets or read from previously captured files. It has a terminal-oriented version called TShark. It offers both live and offline analysis, allowing for either analysing the network traffic in real-time or from captured files in various formats. Wireshark also allows users to use filters and searches on the captured packets, making it a versatile tool with widespread use.

[50] https://www.snort.org/
[51] https://www.cisco.com/c/pt_pt/index.html
[52] https://suricata.io/
[53] https://oisf.net/
[54] https://www.wireshark.org/
[55] https://www.tcpdump.org/

2.2.5.2 Tcpdump

Tcpdump[56] is a packet *Analyser* used in the CLI, displaying the TCP/IP and other packets in the network. Similarly to Wireshark, it is capable of capturing and reading from files. It offers much of the same functionalities as Wireshark. However, not having a GUI makes it a secondary option for some users.

2.2.6 Classifier

Classifiers are capable of classifying the type of traffic they analyse. These classifications can range from identifying the origin of the traffic, such as email, voice, and others, as well as distinguishing between benign and malicious traffic. These tools allow for better security monitoring and threat detection, helping to identify attack patterns in datasets to later be applied in a network.

Previously mentioned tools such as ICSFlowGenerator, HDFEF, NFStream, Snort and Suricata have this capability. FlowTransformer is another tool specifically made to classify traffic which is not available.

2.2.6.1 FlowTransformer

Zhao et al. [34] created **FlowTransformer** to classify traffic from anonymity networks, which are networks that use encryption for the underlying communications and transmissions to hide the user's network identity. For this reason, traditional packet analysis is unable to classify the resulting encrypted traffic. FlowTransformer, to be able to classify the traffic, sets higher weights (values) for important flows, taking into consideration the relation between flows rather than just individual ones, improving the classification. Afterwards, they extract the subsequent features according to these rankings. Finally, it takes already generated flows and classifies them. Therefore, FlowTransformer is a *Classifier* with the capability to extract features.

2.2.7 Other Tools

As mentioned previously, traffic analysis can be conducted not only through flow-based methods but also by directly analysing individual packets. In this regard, in 2022, Farrukh et al. [35] created **payload-byte**[57] that extracts over 1500 features and labels PCAP files, meaning previously captured traffic. The authors found their tools generated good results when compared to other flow-based datasets.

[56] https://www.tcpdump.org/
[57] https://github.com/Yasir-ali-farrukh/Payload-Byte

2.2.8 Findings and Discussion

Based on the information presented in the preceding sections, it is evident that a diverse array of tools is currently in use. Table 2.3 provides an overview of the analysed tools and identifies the categories to which they belong while identifying if it is its main function or merely has that capability. It also identifies the tools that are not public and those that present NIDS capabilities.

Table 2.3: Classification of tools.

Tools	Flow Exporter	Feature Extractor		Flow Collector	Generator	Analyser		Capturer	Classifier
		Packet	Flow			Packet	Flow		
CICFlowMeter	★		★				☆	☆	
LycoSTand	☆		★				☆		
nProbe	★		☆	☆				☆	
Softflowd	★		☆	☆			☆	☆	
YAF	★		☆					☆	
Argus	★		☆	☆			☆	☆	
ICSFlow Generator	☆		★					☆	☆
LiteEX [1]	☆	★	★						
CSTITool [1]	☆		★						
HDFEF [1]	☆	★	★						☆
nfdump	☆			★			☆	☆	
Silk				★			☆		
DOROTHEA					★				
Harpoon					★				
PackETH					★				
Ostinato					★				
D-ITG					★				
NFStream	☆		☆			★	★	☆	☆
BRO-IDS/Zeek [2]			☆			★		☆	
Tranalyzer	☆	☆	☆			☆	★	☆	
Snort [2]						☆		☆	☆
Suricata [2]						☆		☆	☆
Wireshark						☆		★	
Tcpdump						☆		★	
Flow Transformer [1]			☆						★
Payload-Byte		★							☆

[1] non-public tool [2] NIDS capabilities ★ tool's main function ☆ presents capabilities

For *Flow Exporters*, with or without Feature Extraction, there is a myriad of tools that can be used and the number is growing with authors trying to improve the creation of flows. Although not less important, *Flow Collectors* are tools used specifically in certain use cases. For example, in large networks with multiple NetFlow sensors requiring a tool to store all collected flows. As such, current tools have already existed for some time, and no mention of new ones was found.

Generators have a similar disposition, with older tools being more commonly available. Generally, real traffic is preferred for analysis over synthetic traffic. There are, however, authors researching tools to generate synthetic data as closely as possible to real traffic. For instance,

Campazas-Vega et al. [29], through DOROTHEA[58], attempt to improve the generation of traffic by closely mimicking real behaviours. They want to benefit from the advantages of synthetic traffic, such as having a fully controlled environment.

Analysers and *Capturers* have capacities that can often be associated with *Flow Exporters*. This is the case as many *Flow Exporters* have capturing modes to collect packets and then turn them into flows. For *Analyser* capabilities, many *Flow Exporters* allow for analysing the flows after they have been aggregated.

Finally, *Classifiers* are predominantly found in newer tools. This is because as traffic analysis evolves, the necessity for good methods to classify traffic, as benign or malicious, has increased. This classification of traffic, in the form of labels, allows for the creation of good NID datasets, which will be expanded in the next section.

The different categories presented are crucial in the dataset creation workflow. Figure 2.2 illustrates a dataset creation workflow, highlighting the different stages where the categories can be integrated. As presented in the figure, datasets can be created following several paths. They can be a simple packet capture from a network interface or from a *Generator* tool simulating traffic. Still, at the packet level, a dataset can also be created going first through a feature extractor or a *Classifier*, or by passing through these tools in sequence to create a more detailed dataset. At flow level, a dataset can also be created in different ways. The simplest one is when real or synthetic data is captured, and the flows are aggregated in a *Flow Exporter* to directly generate a dataset. Another option is to use a *Flow Collector* to collect features which can be extracted and/or classified to generate a dataset with more information.

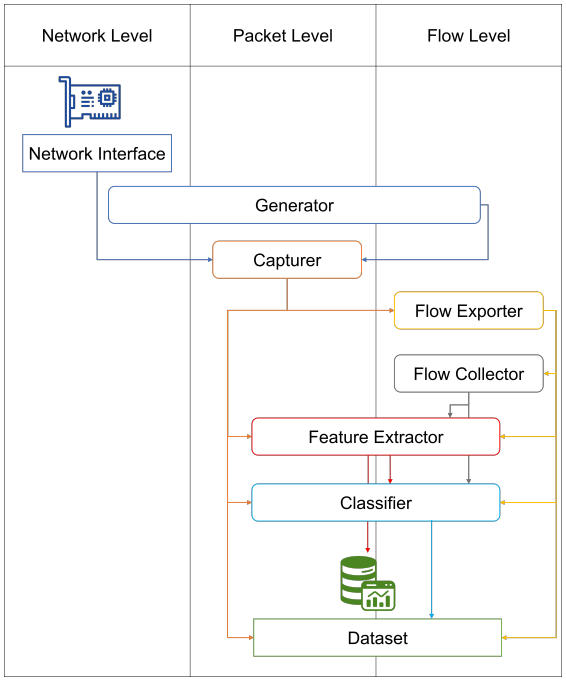
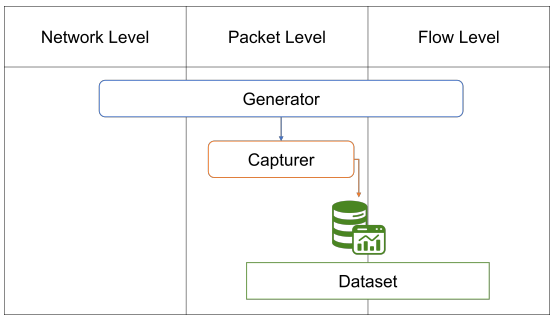


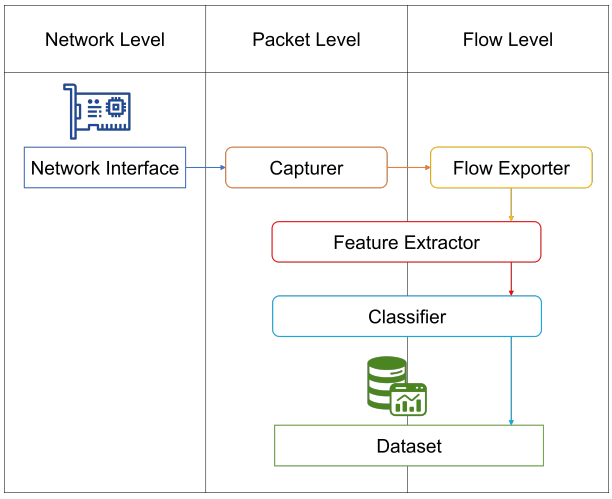
Figure 2.2: Flowchart of dataset creation.

[58] https://seguridad.unileon.es/index.php/DOROTHEA

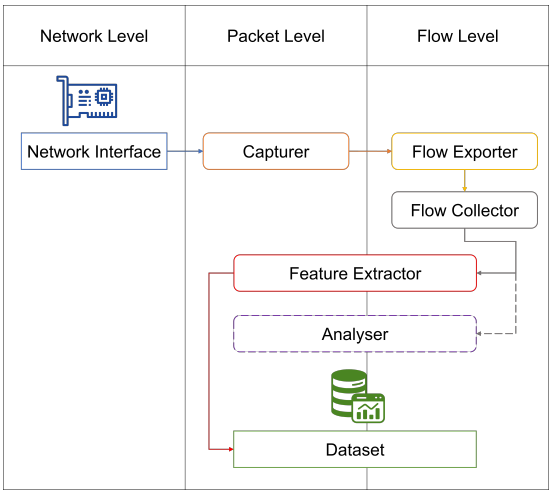
Figure 2.3 provides examples of different paths that can be taken for dataset creation. Figure 2.3a exemplifies a simple dataset creation process. In this case, synthetic data is captured and directly transformed into a dataset. Although a possible process, it does not create a common or useful dataset. Figure 2.3c details the creation of a dataset where the traffic is captured in a network interface and uses a *Flow Exporter* with Feature Extraction to produce flows with features. This will be passed to a *Classifier* to label data as normal or abnormal traffic before adding all the information into the final dataset. Finally, another possible example is presented in Figure 2.3b where a process similar to Figure 2.3c happens, only using a *Flow Collector* as an intermediary. A case like this would most likely happen when there are several *Flow Exporters* in a network. In this example, it is the *Flow Collector* that stores the flows of the whole network. Then it sends the data to extract features in a Feature Extractor, and the dataset is created. In the middle of this process, and if desired, the flows can also be sent to an *Analyser* so that the data can be pre-analysed. This, however, has no implication in the dataset creation process. We may conclude that several combinations of tool categories can be used to create a dataset, it all depends on the objective.



(a)



(b)



(c)

Figure 2.3: (a) Creation of a simple dataset with only synthetic data. (b) Creation of a dataset with flow data, features and labelling. (c) Creation of a dataset with flow data and features, with data that is previously analysed.

2.3 Evolution and Classification of NIDS Datasets

There are a plethora of NIDS datasets that have been created in the last twenty years, these were produced with the tools mentioned in the previous section. These were created through various methodologies, ranging from a simple capture of network interface data in the form of packets to the creation of synthetic data. Datasets may even involve the analysis and classification of flows, culminating in a labelled dataset. Therefore, different datasets can be created using different tools to obtain different results, be that at packet or flow level, labelled or unlabelled. Additionally, studies have been performed on older datasets, providing criteria for researchers to abide by when planning new NIDS datasets, aiming to improve the creation process and the available data to be used by other researchers.

The study performed by Gharib et al. [11] presented an evaluation of older datasets, that have been heavily referenced in the last two decades, to test if a dataset was suitable. Subsequently, the authors proposed 11 criteria for the creation of high-quality datasets, addressing the shortcomings identified in the reference datasets. These criteria included complete network configuration and traffic. The authors considered it necessary to implement a realistic configuration of a network, which includes traffic from all parts of a network, such as a host, router or switch. This traffic can be real, synthetic or a mixture of both. They also considered that a good dataset should have labels, feature sets and metadata to permit its analysis. Besides this, they emphasize the importance of researchers providing complete documentation regarding network interactions, captures, protocols used, and attack diversity when developing a dataset. Finally, anonymity and heterogeneity were found to be required by the authors to provide a good dataset to respect privacy and a more complete process for NID with various sources of traffic.

Kenyon et al. [36] rank popular datasets according to their relative utility and analysis of dataset composition, methodology, maintenance, data quality issues and relevance. Additionally, they provide insights on how to assess datasets for accuracy, scope and currency, meaning evaluating a dataset on the recency of their threats and their deployment. Best practices in dataset design and shortcomings in anomaly detection techniques are also identified.

Based on this study, and as previously mentioned, due to its characteristics, a dataset can be categorized in several ways. There can be a division by whether the collection is flow-based or packet-based. In this division, the packet-based datasets will contain the whole data while the flow-based are comprised of packet data combined into flows, and both can have labelling. Packet-based datasets can have the problem of making it difficult for inspection systems to detect attacks because of their processing overhead, while flow-based datasets can be used to provide a more effective way to detect anomalies in network behaviour [37].

Another form of categorization can be based on the source and nature of the data: real traffic data is actual network traffic from real environments, valuable for accurate analysis but potentially containing sensitive information. Synthetic data is generated by tools simulating network patterns and attack scenarios, useful for controlled testing without security risks. Hybrid

data combines real and synthetic data, leveraging the authenticity of real traffic and the flexibility of synthetic elements to enhance the robustness and performance of Intrusion Detection Systems (IDSs) under various conditions. The advantages of using synthetic traffic come from having an easier definition of ground truth in a dataset, of what is an attack or benign traffic. With synthetic data, the researchers can fully control this aspect at the cost of realism which can lead to model overfitting or an inherent bias [38, 39].

Additionally, other forms of dataset division can be based on their sources, as demonstrated by Ageyev et al. [40] in their survey, where they categorized the datasets into network traffic-based, internet traffic-based, virtual private network-based, and Internet of Things (IoT) traffic-based.

Given the large number of NID datasets found in the literature and their varied uses, we decided to divide them based on their creation date. Accordingly, we define two categories: *Reference datasets* and *Recent datasets*. This division allows us to observe the evolution of dataset creation, as many recent datasets are based on reference datasets. Figure 2.4 illustrates a timeline of the datasets reviewed in this section. *Reference datasets* are depicted in blue, while *Recent* ones are shown in green.

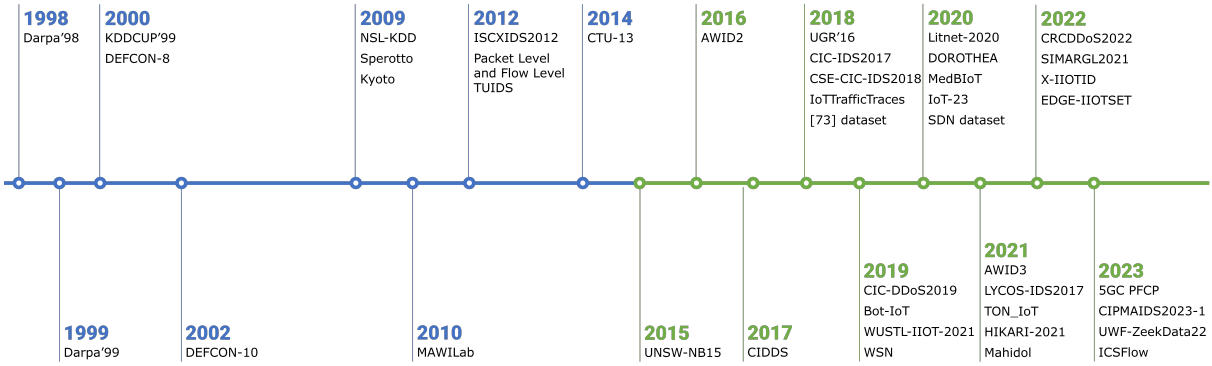


Figure 2.4: Timeline of selected datasets. In blue are the *Reference datasets*, in green, the *Recent datasets*.

2.3.1 Reference Datasets

This section presents datasets that are now considered outdated but still hold importance as foundational resources, often serving as building blocks for further enhancements in dataset development. For example, as Ageyev et al. [40] notes, the CIC[59] family of datasets are partially based on ISCXIDS2012, one of the first datasets created by CIC. Other than those, URG'16 is also partially based on CTU-13. Even NSL-KDD, which is an outdated dataset widely used [41], is based on another outdated dataset, KDD-CUP'99, which is based on an even older dataset, DARPA.

[59] https://www.unb.ca/cic/

2.3.1.1 DARPA datasets (1998 and 1999)

The first efforts in creating a dataset for IDS purposes was done by MIT Lincoln Laboratory[60] which resulted in **DARPA'98**[61] and **DARPA'99**[62]. These datasets consist of synthetic traffic collected during 7 and 5 weeks, respectively, using tcpdump. They present a variety of data and attacks, having served as an important first collection and evaluation of what a dataset for network analysis should include. Since then, these datasets have received increased criticism, but they retain value despite being outdated for NID evaluation. One author who offered some criticism is McHugh [42], in the year 2000, who details an evaluation of DARPA'98 and a brief discussion of DARPA'99. He highlighted issues with their design, execution, and methodology, suggesting they may have produced biased results. A significant point of contention was the use of synthetic data in the datasets. Furthermore, the datasets are now outdated in terms of both attack types and network infrastructure.

2.3.1.2 KDDCUP'99 (2000) & NSL-KDD (2009)

In 2000, Stolfo et al. [43] used the tcpdump files from DARPA'98, to create **KDDCUP'99**. This dataset contains 42 features that are labelled as either normal or attack, using BRO-IDS/Zeek. Each attack from the datasets can be classified into one category of the following four: Denial of Service (DoS), User to Root Attack (U2R), Remote to Local Attack (R2L) or probing attack.

In 2009, Tavallaee et al. [44] presented issues in KDDCUP'99 and proposed a new dataset, **NSL-KDD**, which was created using selected records of KDDCUP'99. The authors initially explain that the dataset contains an excessive number of redundant records, with duplicates present in both the training and testing sets. This redundancy leads to biased results, particularly in frequent records. The authors also recognize that the problems found by McHugh [42] are not fixed in this new dataset. To create NSL-KDD, they removed the redundant records in the training and testing sets, and sampled records where the prediction was correct to create a more challenging data set. Despite this, the problems remained and the threats present in all DARPA-derived datasets are considered outdated.

[60] https://www.ll.mit.edu/
[61] https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset
[62] https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset

2.3.1.3 DEFCON-8 and DEFCON-10 (2000-2002)

Two datasets, **DEFCON-8** and **DEFCON-10**, were generated at the hacker convention known as DEF CON[63] by The shmoo group[64], using traffic collected during the Capture the Flag competition held at that event. These datasets primarily consist of intrusive traffic rather than normal background traffic. DEFCON-8 was produced in 2000 containing Port Scanning and Buffer Overflow attacks. On the other hand, DEFCON-10 was produced in 2002 containing Port Scan and File Transfer Protocol (FTP) by Telnet Protocol attacks, among others. Given its lack of normal background traffic, it is not an efficient NID dataset, but it is useful for studying attacks and alert correlation techniques [36, 40].

2.3.1.4 Sperotto (2009)

In 2009, Sperotto et al. [45] created a labelled flow-based NID dataset, Sperotto, also referred to as Twente, which consists of labelled real traffic. It used tcpdump to record the packets and softflowd to form the flows. This dataset lacks real background traffic, as it comprises malicious traffic from a Honeypot using NetFlow. Because of this, and the lack of diversity of attacks, this dataset is not good for NIDS.

2.3.1.5 Kyoto (2009)

Kyoto was created by Song et al. [46] in 2009 utilizing Honeypots, consequently featuring solely the attacks detected and captured by these Honeypots. It simulates normal traffic solely from Domain Name System (DNS) and mail traffic, lacking a representation of typical network traffic. Moreover, it contains no false positives, which can be counterproductive for NIDS, as they play a role in minimizing alerts. This lack of false positives makes it limited for today's research. Regarding its capture, Kyoto's data was collected over almost 3 years from 2006 to 2009 with the use of several NIDS, namely, Snort. In terms of features, Kyoto has 14 features based on KDDCUP'99 and 10 additional features, all extracted with BRO-IDS/Zeek.

2.3.1.6 MAWILab (2010)

In 2010, Fontugne et al. published a work presenting a new dataset [47] named MAWILab, which was created by capturing real traffic between Japan and the United States on a daily basis for 15 minutes. Its main limitation lies in the brief duration of data collection each day. On the other hand, an advantage is its periodic updates, enabling trend and pattern analysis over time. Notice that its collected data spans from 2001 to the current year, 2024, encompassing various protocols and possible attacks. Notably, only traces since 2007 feature labelled data, denoted as version 1.1, while unlabelled data is marked as version 1.0.

[63] https://defcon.org/
[64] http://cctf.shmoo.com/projects.html

2.3.1.7 ISCXIDS2012 (2012)

The ISCXIDS2012 dataset was developed by Shiravi et al. [48] in 2012 at CIC. It comprises real traffic generated within a controlled test environment featuring simulated attacks and events, such as DoS and Secure Shell (SSH) Brute Force. The traffic was captured and collected with the use of Snort and tcpdump. While this dataset provides flow summaries, it also has several limitations. Notably, the distribution of attack traffic is not based on real-world statistics. Additionally, despite containing a significant portion of web traffic, it lacks Hypertext Transfer Protocol Secure (HTTPS) flow data, which is essential for current standards [36]. Other than ISCXIDS2012, the authors have produced older datasets focusing on topics such as VPN, Tor and Botnet [49].

2.3.1.8 CTU-13 (2014)

CTU-13, which was made available in 2014, is a labelled dataset containing a variety of benign and malicious traffic. It includes live Botnet attacks and flow records aggregated with Argus. The traffic was captured with tcpdump and the labels on the dataset were manually produced based on the IP addresses of the Botnets. Although it is a valuable dataset, its shortcoming comes from not having recent threat attacks.

2.3.2 Recent Datasets

This section introduces datasets designed to overcome the limitations of the previously discussed *Reference datasets*. This includes improving the generation process, whether by capturing traffic from real in-use networks or by improving the extraction of information from flows or packets, and even providing more up-to-date attacks on the datasets.

2.3.2.1 UNSW-NB15 (2015)

In 2015, Moustafa and Slay [50] released UNSW-NB15 as a response to the faults found in the *Reference datasets*, having mentioned KDDCUP'99 and NSL-KDD. One of the faults identified was the limited amount of attacks present in these datasets. Thus, they performed 9 different types, including DoS and Worms. To capture the data of the dataset in packet form, the authors used tcpdump for 31 hours on two separate days, almost 1 month apart. This dataset uses synthetically generated traffic, simulated using IXIA PerfectStorm[65], a traffic *Generator* that emulated real-world network behaviour. With the PCAP files, Argus and BRO-IDS/Zeek were used to obtain the features from the generated network flows.

[65] https://www.keysight.com/us/en/products/network-test/network-test-hardware/perfectstorm.html

2.3.2.2 AWID Datasets (2016 and 2021)

In 2016, Kolias et al. [51] created **AWID2**, a dataset that consists of data from a physical laboratory that simulates Small Office/Home Office (SOHO) infrastructure, covered by a single Access Point (AP) with 10 client stations with some being mobile inside the office and others not. The network also had a mobile attacker that performed various 802.11 attacks. For traffic capturing, the terminal version of Wireshark was used, Tshark, generating 156 packet features.

Then, in 2021, Chatzoglou et al. [52] extended the work performed in AWID2 and created the **AWID3** dataset, focusing on multi-layer and modern attacks. For this, they produced 254 features, with one being for labelling purposes. For this dataset, 16 physical devices and virtual machines were used with one attacker. Wireshark was once again used to collect the traffic generated in the topology.

2.3.2.3 CIDDS Datasets (2017)

In 2017, Ring et al. [53, 54] created the **CIDDS-001** and **CIDDS-002** datasets using a virtualized environment with OpenStack and traffic generated by Python scripts emulating user behaviour. The authors simulated a small business environment with benign and malicious traffic consisting of Brute-Force and DoS attacks, among others. For CIDDS-001, the data was captured for over four weeks within OpenStack. For CIDDS-002, the same process was followed, but the traffic was collected over two weeks.

2.3.2.4 UGR'16 (2018)

In 2018, Maciá-Fernández et al. [2] developed the UGR'16 dataset, constructed using real traffic and featuring up-to-date attacks, such as DoS, Port Scanning, and Botnet traffic. Notably, the dataset considers long-term evolution and traffic periodicity, taking into consideration variations between daytime and nighttime, as well as weekdays and weekends. For this, they captured anonymized NetFlow traces in a tier-3 Internet Service Provider (ISP) for 4 months, presenting it with realistic attack scenarios and labelled traffic. To guarantee privacy and reduce the volume of data, they provide only the NetFlow information. For the capture of the flows, the NetFlow v9 format was used by Cisco's routers.

2.3.2.5 IoTTrafficTraces (2018)

In 2018, Sivanathan et al. [55] created IoTTrafficTraces, an Internet of Things (IoT) dataset from laboratory traffic with 28 IoT devices trying to emulate a smart environment with cameras, lights, plugs, motion sensors, appliances and health monitors for 6 months. This dataset does not have attack traffic. For the collection of the traffic, tcpdump was used and produced 10 features at flow level using the tool Joy[66] to turn PCAP files into flows. Joy is a package from Cisco[67] that captures and analyses flow data permitting network research, forensics and security monitoring.

2.3.2.6 CIC IDS datasets (2018-2019)

An important recent production of NID datasets comes from CIC[68], the creators of ISCX2012. In recent years, they have produced widely known datasets such as CIC-IDS2017 [56], CSE-CIC-IDS2018[69] and CIC-DDoS2019 [57], as well as others that focus on IoT, DNS, NID and Malware. Their goal is to provide reliable and up-to-date datasets and, for this, all were created with real traffic. Interactions between users were simulated in a controlled setting using CICFlowMeter as a collection and feature extraction tool.

For the creation of **CIC-IDS2017**, in 2018, Sharafaldin et al. [56] produced benign traffic, simulating a human, and common attacks, creating the abstract behaviour of 25 users. For 5 days, they collected traffic and presented detailed descriptions of the attacks performed, as well as other information they had previously determined necessary to build a reliable benchmark dataset. With their tool, CICFlowMeter, they extracted more than 80 network flow features. The version of the tool to produce the dataset was impossible to retrieve.

For **CSE-CIC-IDS2018**, in 2019, CIC collaborated with Communications Security Establishment (CSE)[70] to generate a dynamic dataset using a systematic approach. For the production of the dataset, a similar strategy to CIC-IDS2017 was used, with an increase in the number of machines and overall network. This time they simulated a victim organization with 5 departments, 420 machines and 30 servers for 10 days. For the attacks, Version 3 of CICFlowMeter was used to extract more than 80 features. Attacks for the two datasets, CIC-IDS2017 and CIC-IDS2018, included but were not limited to, Distributed Denial of Service (DDoS) and Botnet.

For **CIC-DDoS2019**, in 2019, Sharafaldin et al. [57] created a dataset that focused on DDoS to aid in detecting different types of DDoS. Again using CICFlowMeter, version 3, traffic was collected in a similar method to CIC-IDS2017, with a simulation of 25 users over 2 days. Resulting in another 80-plus features extracted. Although their datasets are commonly used, there have been reported issues regarding their creation, as mentioned in Section 2.2.1.1, mainly due to CICFlowMeter.

[66] https://github.com/cisco/joy
[67] https://www.cisco.com/c/pt_pt/index.html
[68] https://www.unb.ca/cic/
[69] https://www.unb.ca/cic/datasets/ids-2018.html
[70] https://www.cse-cst.gc.ca/en

In 2018, Panigrahi and Borah [58] detailed the first shortcomings of the CICIDS2017. Key issues included missing values in the dataset, affecting both the “label” and instances with incomplete information. Additionally, a high-class imbalance was identified, with 83.43% of the dataset being benign traffic and certain attacks having a low appearance rate.

In 2021, Engelen et al. [22] published an analysis of CIC-IDS2017 and CICFlowMeter. In their investigation, they found errors in the attack simulation, the extraction of features, the labelling and the benchmarking of the dataset. A patch for the CICFlowMeter tool was provided, addressing identified errors that led to problems with dataset generation. CSE-CIC-IDS2018 was also found to present the same errors in its flow construction. After the publication of this work, the GitHub page has been updated to a new version, namely, version 4 of the tool[71].

In 2021 and 2022, Rosay et al. [23, 24] discovered problems in the CSV files and CIC-FlowMeter. These issues were feature duplication, feature miscalculations, wrong protocol detection, inadequate TCP session termination and labelling issues. This allowed the authors to conclude that the flows were inaccurate due to the incorrect protocol detection, which led to the miscalculation of features. Beyond that, four artefacts were found in the CSV files that showed that the source of the problems was the tool. Consequently, other datasets generated by CICFlowMeter were expected to contain similar issues.

In 2023, Lanvin et al. [25] discovered more problems with CIC-IDS2017 in addition to those that had already been discovered and discussed. These included Port Scan attacks incorrectly labelled and duplicated traffic that resulted in feature extraction and labelling issues. In their work, they use the updated tool by Engelen et al. [22] and discard a solution provided by Rosay et al. [23] for missing some packets which led to incorrect statistics. Lanvin et al. [25] added onto the patch created by Engelen et al. Lanvin et al. [25] to verify the packet’s order. This was done to avoid an issue with packet misordering that originated incorrect flows. Other contributions include the use of editing tools such as editcap, a CLI utility of Wireshark, to find and remove duplicated traffic as well as relabelling flows as port scan attacks.

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

2.3.2.7 LYCOS-IDS2017 (2021)

In 2021, Rosay et al. [23] introduced a dataset named LYCOS-IDS2017 together with their tool, LycosSTand. These served as a solution to overcome the issues uncovered about CIC's[72] datasets and CICFlowMeter. The dataset uses the PCAP files from CIC-IDS2017, but aggregates the flows using their new tool. Additionally, as CIC uses a proprietary package to generate the labelling of their datasets, the authors developed a Python solution to label the traffic of their newly generated dataset. To correctly label the dataset, the authors observed the packets using Wireshark and the information provided by CIC on their website about the timings of attacks. As they were limited to information provided by CIC, they could not keep the traffic from Thursday afternoon of the original CIC-IDS2017 dataset, because they could not label the data based on the provided information. This resulted in a dataset with fewer instances. In their experiments, using ML algorithms, they obtained better performance with LYCOS-IDS2017 than CIC-IDS2017.

2.3.2.8 Other Datasets using CICFlowMeter (2022-2023)

Even with the reported flaws in CICFlowMeter, there have been other authors who used this tool to generate their datasets using their own PCAP files.

In 2022, Hadi et al. [59] produced **CRCDDoS2022** as a more comprehensive dataset for DDoS attacks. They produced different types of attack traffic from simulators and collected the information using tcpdump. Afterwards, CICFlowMeter was utilized to obtain features from the raw data collected, obtaining over 80 features.

In the following year, two more datasets were generated using CICFlowMeter for the extraction of features: **5GC PFCP** and **CIPMAIDS2023-1**. Amponis et al. [60] generated 5GC PFCP NID dataset that focused on the security issues related to Packet Forwarding Control Protocol (PFCP). To produce flow statistics for the TCP/IP protocol, they utilized CICFlowMeter. For the PFCP flow statistics, they developed a custom PFCP flow *Generator*. This dataset respects the 11 criteria proposed by Gharib et al. [11].

On the other hand, Ali et al. [61] produced CIPMAIDS2023-1. They simulated a network in GNS3[73] and performed various attacks (e.g. DoS and Web Attacks) from different machines, collecting the traffic in Wireshark and forwarding the traffic to CICFlowMeter for feature extraction. In their work, they mention the issues that the CICIDS2017 dataset presented but make no mention of the issues found in CICFlowMeter. Therefore, they produced this new dataset in an attempt to overcome the original dataset limitations, but the issues that CICFlowMeter presents were not addressed.

[72] https://www.unb.ca/cic/
[73] https://www.gns3.com/

2.3.2.9 Bot-IoT (2019)

In 2019, Koroniotis et al. [62] created Bot-IoT to study network behaviour in an IoT environment, including thermostats, motion-activated lights, a smart fridge and a weather station. The dataset contains both IoT and normal network user traffic, with various attack types used to simulate Botnet scenarios, with no actual malware being used. It is an unbalanced dataset since it has more records for some attacks than for others.

2.3.2.10 WUSTL-IIOT-2021 (2019)

In 2019, Zolanvari et al. [63] created **WUSTL-IIOT-2021**, a dataset of emulated data from Industrial Internet of Things (IIoT) traffic, spanning a total of 53 hours. The emulation occurred by using a real system monitoring water level and clarity in a storage tank. Traffic collection was done with Wireshark, while Argus was used for flow creation and feature generation, resulting in 42 features, including one label. Three attacks were executed in the testbed: Backdoor Virus, Command Injection and Structured Query Language (SQL) Injection. Additionally, the authors also identified 6 features recommended for removal to prevent the ML model from being exposed to the type of attack performed.

2.3.2.11 Litnet-2020 (2020)

In 2020, Damasevicius et al. [39] created their dataset, Litnet-2020, with traffic collected during 10 months with the nfcap tool. A mix of Nfsen, MeSequel and Python scripts was used to extract features. With this dataset, the authors' main objective was to represent real-world network traffic and attacks on real infrastructure. For this, the dataset contains traffic country-wide, from Lithuania, with servers in 4 different cities. The dataset has 12 attack types, including DoS and Port Scannings.

2.3.2.12 DOROTHEA (2020)

As mentioned in Section 2.2.3.1 Campazas-Vega et al. [29] created a traffic *Generator* tool, DOROTHEA, and with it generated a new dataset with the same name. The tool was created to generate traffic, so the dataset consists of synthetic data with benign and malicious traffic. Alongside Python scripts, the authors created a training dataset, D1, and a testing dataset, D2, for fitting classification. The main difference between the two datasets is in the attacks, where D1 had continuous Port Scanning and D2 had slow Port Scanning attacks.

2.3.2.13 MedBIoT (2020)

In 2020, Guerra-Manzanares et al. [64] created a dataset, MedBIoT, focusing on IoT with normal (real and emulated) and malicious network traffic from Botnet in a network with 83 IoT devices. The network topology comprised of 2 connected networks, with direct internet connection for initial device setup. The monitoring network was responsible for the collection of data, using tcpdump and Splunk for its processing and labelling. The final network, the IoT Local Area Network (LAN), consisted of the environment where malware was contained with virtual IoT devices to generate the network behaviour with the use of Docker[74].

2.3.2.14 IoT-23 (2020)

In 2020, Garcia et al. [65] introduced IoT-23, a dataset that contains 20 captures of malware and 3 captures of benign IoT traffic. It was collected from IoT devices spanning the years 2018 to 2019 at the Stratosphere Laboratory[75] in CTU University, Czech Republic, with research funded by Avast Software[76]. The 23 captures were divided into 20 PCAP files, with Malware executed on a Raspberry Pi and the benign captures in a Philips HUE smart LED lamp, an Amazon Echo home intelligent personal assistant and a Somfy smart door lock. The use of these real devices allowed the authors to capture and analyse real network behaviour. For the generation of the dataset, an unspecified software captured the traffic, which was subsequently turned into flows and analysed by BRO-IDS/Zeek. Flow data labelling was done with the university's labelling software, Flaber[77], a python tool that reads BRO-IDS/Zeek files, analyses columns and inserts labels.

2.3.2.15 TON_IoT (2021)

In 2021, Moustafa [66] created TON_IoT, a group of four IoT datasets comprised of telemetry data of IoT services from a controlled environment with normal and malicious traffic, such as DoS/DDoS and Ransomware, among others. The dataset also includes the logs from the operating systems. The problem that these datasets present is insufficient network features, having only network addresses, the amount of transferred traffic and some protocol-specific features, originating a lack of understanding of the network behaviour during the attacks [27].

[74] https://www.docker.com/
[75] https://www.stratosphereips.org/about-stratosphere
[76] https://www.avast.com/pt-pt/index#pc
[77] https://github.com/stratosphereips/flaber

2.3.2.16 HIKARI-2021 (2021)

In 2021, Ferriyan et al. [38] generated an encrypted synthetic dataset, HIKARI-2021, with attack (e.g. Bruteforce and Probing) and benign traffic. They begin by collecting the generated synthetic traffic, in PCAP format using tcpdump, followed by processing the packets collected to anonymize it and extract the features with BRO-IDS/Zeek. The authors created this dataset with new requirements for dataset creation such as needing anonymization, payload, ground-truth, encryption, and a practical method to implement it. For the features generated, Hikari-2021 bases its features on CICIDS2017, with extra ones to reference the flow, such as IP addresses and ports.

2.3.2.17 SIMARGL2021 (2022)

In 2022, Komisarek et al. [67] created SIMARGL2021, a dataset with flow-based features that could be optimal for real-time detection. To collect the dataset, nProbe was used with NetFlow version 9, for 8 days, creating 20 CSV files. Their data collection workflow consisted of the use of nProbe, followed by Kafka[78]. This enabled data streaming to process it with ML algorithms, labelling each NetFlow frame. Finally, it gets sent to Elasticsearch[79] to combine features and allow for data analysis. The malicious traffic is divided into three types: Reconnaissance, DoS and Botnet attacks.

2.3.2.18 X-IIOTID (2022)

In 2022, Al-Hawawreh et al. [68] created X-IIoTID, a labelled dataset based on IIoT, which was created using their realistic IIoT security testbed, Brown-IIoTbed, implemented in a IoT lab in the researcher's university. The testbed integrates new IoT technologies with legacy industrial devices. The objective of its creation was to improve NID in a complex IIoT network. It has a total of 18 types of intrusions within the dataset ranging from Fuzzers to Brute Force.

2.3.2.19 EDGE-IIOTSET (2022)

In 2022, Ferrag et al. [69] presented EDGE-IIOTSET, a new dataset with realistic IoT and IIoT traffic. The traffic was obtained using various IoT devices, such as sensors for temperature, humidity, pH, and others. The authors produced 61 features after using BRO-IDS/Zeek and Tshark, the terminal version of Wireshark. The attacks range from DoS/DDoS, to Man-in-the-Middle (MitM) attacks, among others.

[78] https://kafka.apache.org/
[79] https://www.elastic.co/pt/elasticsearch

2.3.2.20 UWF-ZeekData22 (2023)

In 2023, Bagui et al. [70] developed UWF-ZeekData22, using BRO-IDS/Zeek to capture the network frames and saved them in logs and PCAPs. They labelled the dataset using the MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) [71] framework to improve effectiveness against new threats. In total, the dataset contains 14 MITRE ATT&CK attacks, including Reconnaissance, Privilege Escalation, and others, collected over 16 weeks.

2.3.2.21 ICSFlow (2023)

In 2023, Dehlaghi-Ghadim et al. [27], the creators of ICSFlowGenerator, developed ICSFlow for evaluating ML-based NIDSs in industrial systems. Simulated traffic from an Industrial Control System (ICS) controlling factory equipment, that fills empty water bottles from a tank, virtualized on Docker, was captured with tcpdump, labelled with MITRE ATT&CK using Python scripts, and processed with ICSFlowGenerator to aggregate into flows and generate the features. From the MITRE ATT&CK attacks the datasets uses Reconnaissance, Replay, DDoS and MitM.

2.3.2.22 Other Datasets

Besides the datasets previously presented, there were others not accessible, namely the ones described next. For this reason, their description is exclusively based on the papers that introduce them.

In 2012, Gogoi et al. [72] proposed two datasets, **Packet Level** and **Flow Level TUIDS**, with real traffic to improve the existing datasets primarily consisting of unrealistic simulated traffic. Although the datasets were mentioned to be public, retrieval was not possible. The datasets were generated in an isolated network with malicious traffic from DoS and Probe attacks. The packet-level dataset was captured with gulp[80] and the features extracted with tcptrace. For the flow dataset, features were extracted using C programs with flow collection by nfdump after an unspecified *Flow Exporter* aggregated them.

In 2018, Uramova et al. [73] created a dataset with DoS and Botnet attacks having the findings by the authors Gharib et al. [11] as criteria, which Uramova et al. [73] were unable to meet. In specific, they were unable to correspond to the extensive protocols required by the criteria, the anonymity and the metadata necessary.

In 2019, Nivaashini and Thangaraj [74] created an untitled dataset referred to here as **Wireless Sensor Networks (WSN)** as it is one of the main subjects of the paper. The dataset is composed of real IoT data and unidentified attack traffic captured by Wireshark, and features are extracted with the use of scripts.

In 2020, Kaan Sarica and Angin [75] introduced an unnamed dataset focusing on Software Defined Networking (SDN) traffic, here referred to as **SDN** dataset. Their goal was to create an IoT dataset managed by SDN. They utilized a similar topology to Bot-IoT, generating and

[80] https://gulpjs.com/

collecting traffic with and without malicious traffic, including DoS/DDoS and Fuzzing. They simulated their network using Mininet[81], a tool for generating synthetic SDN traffic. To facilitate flow and feature generation, the authors developed a custom application for both tasks, including flow labelling.

In 2021, Wilailux and Ngamsuriyaroj [37] presented a framework intending to generate bi-directional traffic. Referred here as **Mahidol**, the resulting dataset combined both replicated session traffic and attack traffic generated by the framework. They claim that the dataset's improvement lies in diverse background traffic, representing various application protocols like DNS, FTP, Hypertext Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP) and SSH. Furthermore, their malicious traffic generation of Scanning and SSH Brute Force uses proportionate selection, where each attack type is assigned a fitness score. This means that a higher score increases the chance of the attack happening.

2.3.3 Findings and Discussion

The evolution of available datasets has been continuously expanding to accompany the growth of network communications, and the increase and evolution of cyber threats. Whether utilizing packets or flows, real or synthetic traffic, the significance of diverse and high-quality datasets cannot be overstated. Table 2.4 provides a summary of the information previously presented regarding the various datasets.

As it is possible to observe, there has been an overall increase in the development of datasets and the use of flow data for their creation, as well as the use of labelled data. It is also possible to note a pattern of the type of tools used while aiding in creating these datasets. *Capturers* for the most part are always used. This is because authors choose to capture new data that can be real or created with the use of *Generators*. After that process, *Flow Extractors* and *Feature Extractors* are used with resource to a *Classifier* to label the data, although this kind of tool is not specified many times or is done with scripts. Another important detail to note is that although there is a preference for real data, many authors still choose to use synthetic data or semi-realistic data to generate these datasets. This is mainly because of the challenges that using real traffic poses, such as a higher monetary cost to set up and maintain the network, and a lack of easily reconfigurable and scalable topology, being limited to the physical devices being used. Other challenges include the privacy and anonymization of the collected data as real traffic contains sensitive information both in the form of IP addresses and the payload. Ensuring anonymity without decreasing the usefulness of the dataset is complex with many opting for removing the payload entirely which is necessary for certain mechanisms such as deep packet inspection and thus reducing its usability [11]. Furthermore, using real traffic generates massive amounts of data which can be challenging to store, manage and process efficiently thus being resource intensive and difficulting the process of labelling and establishing ground truth. This is because these types of scenarios hinder experts in determining what traffic is benign or malicious [38].

[81] https://mininet.org/

Table 2.4: Tools used by authors leading to the creation of each dataset.

Datasets	Year	Tools			Type	Traffic	Label	Availability
		Generator	Capturer	Flow/Feature				
Darpa'98	1998	Simulated Network	tcpdump	None	P	S	NO	MIT
Darpa'99	1999	Simulated Network	tcpdump	None	P	S	NO	MIT
KDDCUP'99	1999	DARPA'98 dataset	DARPA'98 dataset	BRO-IDS/Zeek	P	S	YES	UCI
DEFCON-8	2000	None	Unknown	None	P	R	NO	Not Found
Sperotto	2009	None	tcpdump	Softflowd	F	R	YES	Not Found
Kyoto	2009	None	Snort	BRO-IDS/Zeek	P	H	YES	Takakura
NSL-KDD	2009	based onKDDCUP'99 dataset	based onKDDCUP'99 dataset	based onKDDCUP'99 dataset	P	S	YES	UNB
MAWILab	2010	None	Not Specified	Not Specified	P	R	YES[6]	Fukuda
ISCXIDS2012	2012	None	Snort & tcpdump	Unknown	F	R	YES	UNB [1]
Packet Level TUIDS	2012	None	Gulp	tcptrace	P	R	YES	Non-public
Flow Level TUIDS	2012	None	Gulp	C programs	F	R	YES	Non-public
CTU-13	2014	None	tcpdump	Argus & BRO-IDS/Zeek	F	R	YES	Stratosphere
UNSW-NB15	2015	IXIA PerfectStorm	tcpdump	Argus & BRO-IDS/Zeek	F	S	YES	UNSW
AWID2	2016	None	Wireshark	None	P	R	YES	Aegean[4]
CIDDS Datasets	2017	OpenStack	OpenStack	Not Specified	F	S	YES	HS-Coburg
CIC-IDS2017	2018	None	CICFlowMeter	CICFlowMeter	F	R[2]	YES	UNB
UGR'16	2018	None	NetFlow v9 Collectors	nfdump & softflowd	F	R	YES	UGR
IoTTrafficTraces	2018	None	tcpdump	Joy	F	R	NO[5]	UNSW
CSE-CIC-IDS2018	2018	None	CICFlowMeter	CICFlowMeter	F	R[2]	YES	UNB
[73] dataset	2018	Unknown	Moloch	Moloch, Suricata and EveBox	F	S	YES	Non-public
CIC-DDoS2019	2019	None	CICFlowMeter	CICFlowMeter	F	R[2]	YES	UNB
WSN	2019	None	Wireshark	Scripts	P	R	YES	Non-public
Bot-IoT	2019	Ostinato & Node-red	Wireshark	Argus	F	S	YES	UNSW
WUSTL-IIOT-2021	2021	None	Wireshark	Argus	F	R	YES	IEEE
Litnet-2020	2020	None	nfcapd (nfdump)	Nfsen (nfdump), MeSequel & Python scripts	F	R	YES	Litnet
DOROTHEA	2020	DOROTHEA, Python Scripts	DOROTHEA	DOROTHEA	F	S	YES	Unileon
SDN	2020	Mininet	Non-specified	Custom Application	F	S	YES	Non-public
MedBIoT	2020	Virtual environment with Docker	tcpdump	Splunk	P	H	YES	TalTech
IoT-23	2020	None	Unknown	BRO-IDS/Zeek	F	R	YES	Zenodo
AWID3	2021	None	Wireshark	None	P	R	YES	Aegean[4]
TON_IoT	2021	Ostinato	Unknown	BRO-IDS/Zeek	F	S	YES	UNSW
LYCOS-IDS2017	2021	None	based on CIC-IDS2017	LycoSTand	F	R	YES	Le Mans Université
HIKARI-2021	2021	Python Scripts	tcpdump	BRO-IDS/Zeek & Python Tools	F	S	YES	Zenodo
Madihol	2021	Not Specified	Not Specified	Not Specified	F	S	NO[3]	Non-public
CRCDDoS2022	2022	HOIC, Slowloris, DDoSIM, Mass, Golden eye, Bonesi	tcpdump	CICFlowMeter	F	S	YES	GitHub
SIMARGL2021	2022	None	nProbe	Apache Kafka & ElasticSearch	F	R	YES	Kaggle
X-IIOTID	2022	None	Wireshark	BRO-IDS/Zeek	P & F	R[2]	YES	Kaggle
EDGE-IIOTSET	2022	None	Wireshark	BRO-IDS/Zeek	F	R	YES	IEEE
5GC PFCP	2023	5G Testbed	Wireshark	CICFlowMeter & custom PFCP flow generator	F	S	YES	IEEE
CIPMAIDS2023-1	2023	GNS3	Wireshark	CICFlowMeter	F	S	YES	GitHub
UWF-ZeekData22	2023	None	BRO-IDS/Zeek	BRO-IDS/Zeek & MITRE ATT&CK	F	R	YES	UWF
ICSFlow	2023	Virtual environment with Docker	tcpdump	ICSFlow Generator	F	S	YES	GitHub

[1] Official link but no access to the dataset.
[2] Created in a testbed with realistic behaviour.
[3] Paper does not refer if the label exists or not, so it was considered that it does not.
[4] A form must be filled out to ask permission to access the dataset.
[5] Authors mention labelling using the Weka tool but do not provide labels in the public files.
[6] There are two versions, one with labels and one without.
P - packet-level features; F - flow-level features.
S - synthetic traffic; R - real traffic; H - hybrid traffic.

2.4 Impact of Dataset Features in NIDS

In the context of NID, the development of good and effective ML-based systems hinge on the datasets used for training, testing and validation. This section discusses the impact of dataset features and why it is necessary to carefully define them. The effectiveness of any NIDS is intrinsically tied to its capacity to differentiate between normal network behaviour and malicious activity. To achieve this, an ML-based NIDS must be trained on datasets that mirror real-world network traffic to benefit from ML. Features play a vital role in capturing essential network information, facilitating the detection of potentially threatening traffic types within a network. For instance, in the review of Maciá-Fernández et al. [2], the authors underscored the importance of crafting datasets for NID that not only incorporate suitable features tailored to the specific type of NID being trained and tested, but also prioritize providing data in raw network capture format. This approach allows researchers to generate features tailored to their specific requirements.

Regarding the impact of creating features, Ali et al. [76] conducted a review on DDoS detection using active and idle features produced by the outdated version of CICFlowMeter[82] and the most recent one (version 4). This latter version corrected some errors in the formation of flows which affected the creation of the features. The authors found that the erroneous flow formation in the original version resulted in misidentification of attacks, such as Neptune (a SYN flood attack), consequently lowering the accuracy and F1-scores of the Active and Idle features. In contrast, the newer version demonstrated higher accuracy, F1-scores, sensitivity, specificity, and precision, indicating substantial enhancements in detection performance.

Meanwhile, Muralidharan et al. [77] aimed to build a monitoring application that could classify traffic into 7 different types. Through their exploration, they discovered that they could classify network traffic using time-based features without violating data protection laws. They specifically utilized 23 features of the 84 produced by CICFlowMeter and achieved an 80% accuracy with their model. The authors noticed this by using a Random Forest Classifier to identify feature impact and found that features such as flow bytes per second, duration of flow, and flow packets per second, hold more impact when compared to other features.

Furthermore, Sarhan et al. [78] strived to obtain a standard set of features to allow for a more critical and rigorous evaluation of ML-based NIDS. Primarily, the authors found that a main issue of not having a standard is the extraction of irrelevant features. They also found the limited ability to evaluate ML models in regard to specific features across different datasets. Finally, the lack of a universal dataset that contains network data flows collected over multiple network environments was also noted. After analysing the features of different popular datasets and using NetFlow v9 features, the authors obtained 43 flow features that provide general flow statistics and statistics on protocols. When testing their results, the authors found that their 43 features achieved higher classification performance (F1-score) than the ones in the datasets they used [10, 78].

[82] https://www.unb.ca/cic/research/applications.html

2.4.1 Findings and Discussion

Even if features are impactful and there has been an effort for their standardization, datasets still present different features that result from the tools used for their extraction, as well as from the choice the authors made on which features are important to present. Table 2.5 groups the features of the different datasets previously described to provide an overview of their distribution. It is important to note the following. The datasets DEFCON, Sperotto, [73] dataset, WSN and ISCXIDS2012 were not possible to retrieve and, as such, it was impossible to review them. The dataset DOROTHEA and its features were possible to retrieve, but a lack of documentation has not allowed the grouping of its features. The same happened to the AWID datasets. For Litnet-2020, 49 features are NetFlow specific, but the authors do not mention which ones were used. Furthermore, the documentation is not clear regarding all the remaining features. For this reason, none of these datasets were added to Table 2.5. Regarding UWF-ZeekData22, several different features were generated with BRO-IDS/Zeek across 16 files, so it was also decided not to include it in that table.

It is possible to observe that features hold a significant impact when accessing the quality of a dataset, but not only that. Different features allow for the accomplishment of different objectives. In some cases, certain features are necessary for a particular network study, such as the example of the 5GC PFCP dataset that requires PFCP features that other datasets do not [60]. This is because the application of the datasets varies, for example, research on web-related traffic benefits from HTTP-specifc features, but for research on IIoT this is not the case. Nevertheless, certain features are always important, such as the network identifiers, packet statistics and labels.

A noticeable discrepancy among the datasets is the variance of the total number of dataset features. Another is the lack of standardization of features. Each author varies the features they choose to extract, or are chosen according to the tool they use. The most commonly observed features are network identifiers, packet statistics, time statistics (although from dataset to dataset the specific ones vary), flow statistics and flags, mainly TCP flags. The least commonly observed are features regarding the header information, subflows, specific protocols and verifiers. Labels also vary according to whether their authors decided to classify traffic as benign or malicious, or any other information they find necessary.

Table 2.5: Group of features by dataset.

Datasets	Feature Extraction	T	NI	PS	TS	F	H	S	FS	L	P	V	Total
KDDCUP'99	BRO-IDS/Zeek	P	2	23	1	0	0	0	0	1	0	15	42
Kyoto	BRO-IDS/Zeek	P	5	11	3	0	0	0	0	1	0	4	24
MAWILab	Not Specified	P	5	0	0	0	0	0	0	5	0	0	10
NSL-KDD	based on KDDCUP'99	P	2	23	1	0	0	0	0	1	0	15	42
Packet Level TUIDS	tcptrace	P	6	34	1	6	1	0	0	1	0	2	51[3]
Flow Level TUIDS	C programs	F	6	3	5	6	0	0	4	1	0	0	25[3]
CTU-13	Argus & BRO-IDS/Zeek	F	6	6	2	0	0	0	0	1	0	0	15
UNSW-NB15	Argus & BRO-IDS/Zeek	F	6	15	14	0	0	0	2	2	0	10	49
CIDDS Datasets	Not Specified	F	5	2	2	1	0	0	0	4	0	0	14
IoTTrafficTraces	Joy	F	8	1	1	0	0	0	0	0	0	0	10
CIC Datasets	CICFlowMeter	F	7	33	17	12	2	4	8	1	0	0	84
UGR'16 [2]	Nfdump & Softflowd	F	6	2	0	1	0	0	3	1	0	0	13
Bot-IoT	Argus	F	8	16	12	2	0	0	5	3	0	0	46
WUSTL-IIOT-2021	Argus	F	7	19	17	0	0	0	2	1	0	2	48[4]
SDN	Custom Application	F	9	14	4	0	0	0	4	2	0	0	33
MedBIoT	Splunk	P	0	0	100	0	0	0	0	0	0	0	100
IoT-23	BRO-IDS/Zeek	F	9	9	2	0	0	0	0	1	0	0	21
TON_IoT	BRO-IDS/Zeek	F	6	8	2	0	0	0	0	5	25	0	46
LYCOS-IDS2017	LycoSTand	F	7	26	19	12	4	4	8	1	0	2	83
HIKARI-2021	BRO-IDS/Zeek	F	5	24	20	10	6	4	15	2	0	0	86
Madihol	Not Specified	F	3	13	4	0	0	0	0	0	0	0	20
CRCDDoS2022	CICFlowMeter	F	6	33	17	12	2	4	8	1	0	0	83 [1]
SIMARGL2021	Apache Kafka	F	6	17	4	2	0	0	1	2	0	0	32
X-IIOTID	BRO-IDS/Zeek; Batch & Python Scripts; OSSEC	P & F	8	16	25	5	0	0	0	4	0	9	68
EDGE-IIOTSET	BRO-IDS/Zeek	F	9	7	0	8	0	0	0	2	35	2	63
5GC PFCP	Custom PFCP Python Parser	F	6	5	1	0	0	0	0	1	23	0	36
CIPMAIDS2023-1	CICFlowMeter	F	7	33	17	12	2	4	8	1	0	0	84
ICSFlow	ICSFlowGenerator	F	3	26	9	12	0	0	0	4	0	0	54
[10, 78] Feature Standard	nProbe	F	7	14	7	3	0	0	6	0	0	6	43

[1] Difference from other datasets using CICFlowMeter comes from the lack of flow ID in Network Identifiers.
[2] The version considered was the CSV format with labelling.
[3] Label feature was added because authors mention their existence but do not indicate it on the feature table.
[4] Dataset has a total of 48 features, but the authors recommend removing 6 of them to not expose the type of the attack to the ML model.

Type of feature (T) - P indicates packet-level features; F indicates flow-level features.
Network Identifiers (NI) - Attributes that contain the identifiers of a flow, such as flow ID, source and destination of IP, port, protocol, and timestamp. Software-specific features, like Argus sequence number, also get considered as an identifier.
Packet Statistics (PS) - Attributes that contain the packet statistics, such as counters, lengths and averages of packets.
Time Statistics (TS) - Attributes related to time spent both in forward and backward direction.
Flags (F) - Attributes that contain counters of flags, such as FIN, SYN, or CWE.
Header Information (H) - Attributes that contain information about the header.
Subflows (S) - Attributes related to subflows.
Flow Statistics (FS) - Attributes with statistics related to the flow.
Labels (L) - Attributes with labels to classify the flow.
Protocol (P) - Attributes that contain protocol-specific information, such as PFCP or HTTP.
Verifiers (V) - Attributes that attribute values, such as 0 or 1 or the result of a calculation, as verifiers of certain conditions.

2.5 Chapter Remarks

This chapter answered the research questions using the PRISMA standard and a snowballing process. For the main research question, it was found that several flow-based tools exist that interact with other tools to generate datasets and overall allow for network traffic analysis. These tools have been in use for almost two decades with increasing development by the scientific community for more accurate tools for network traffic analysis. Answering the first research question, the most popular tool found was CICFlowMeter but other alternatives to improve upon its flaws have been surging, such as LycoSTand or ICSFlowGenerator. Furthermore, other tools have been developed that allow flow generation, analysis and classification, such as DOROTHEA, Tranalyzer and Flow Transformer. It is the combined use of these new and older tools that allows for the creation of new datasets that improve systems that aid in the overall analysis of networks.

Regarding research question number two, it was found that errors in building these tools result in datasets with faulty statistics that in turn obtain worse results in identifying attacks that a network may suffer. In the same analysis, for the third research question, it was possible to assert that different features will be better equipt for certain research. While some features remain necessary, such as flow identifying features, for instances, IP addresses and ports, or behavioural features, for example, flow duration and inter-arrival times between flows that allow detecting irregularities. Others depend on the objectives of the dataset produced, for example, the requirements to study web-related traffic are different from IIoT traffic. Moreover, for real-time analysis, time-based features are not possible to obtain and so would not be used, but for identifying certain services, time-based features are necessary.

Finally, for the final research question, the limitation of these tools relates to false positives and false negatives that are the result of what was previously mentioned. A tool that generates flows and features incorrectly will, in turn, create more false results. Not only that, but the combination of features used affects accuracy, not always needing as many features as these tools are extracting.

Chapter 3

HERA

This chapter details the development of the tool **H**olistic n**E**twork featu**R**es **A**ggregator (HERA) that was created taking into consideration the findings of the systematic review of Section 2.

HERA is a tool developed in Python using Jupyter Notebook, capable of leveraging the *Flow Exporter*, Argus, to produce flows and extract features. HERA allows the user to produce datasets with or without labelling to use them for Network Intrusion Detection (NID) research. It provides an alternative to dataset creation, motivated by the problems discovered with the popular tool, CICFlowMeter, and the errors in the datasets produced by using it. Table 3.1 lists the versions of the software and programming language used in constructing HERA.

Table 3.1: Versions of the software and programming language used to implement the tool.

Function	**Program**	**Version**
Development environment	VirtualBox	7.0.14
Development environment	Ubuntu	23.10
Development environment	Jupyter Notebook	6.4.12
Programming Language	Python	3.11.6

The tool was designed to allow users to utilise either Packet Capture (PCAP) files or HERA flow files, that consist of *argus* data, to generate datasets. These datasets can be created by customising fields such as flow intervals for the exportation of flows and subsequently selecting specific features to be extracted. Furthermore, a labelled version of the dataset can be generated.

To facilitate the use of HERA, the tool is divided into four main components: workspace definition, HERA flow file generation, dataset creation, and dataset labelling. In the workspace definition component, the user is prompted to input path locations for the PCAP files, the HERA flow files and the Comma-Separated Values (CSV) files that will be generated. In the HERA flow file generation component, which can be optional if the files already exist, the user configures flow definition fields and the tool generates the flows. For the dataset creation component,

HERA prompts the user to select features for the dataset and generates the CSV file. Finally, for the dataset labelling component, the tool requires a ground-truth file's location to label the previously generated dataset.

In summary, HERA's workflow can be outlined by the following operations from the tool and the user:

1. Import the required libraries.
2. Workspace definition:
   (a) Select the path for the PCAP files.
   (b) Select the path for the HERA flow files.
   (c) Select the path for the CSV files.
3. HERA's flow file generation (optional if these files already exist):
   (a) Select arguments for generating the HERA flow file to determine which extra information to include.
   (b) Define the interval between flows in seconds.
   (c) Generate the HERA flow files.
4. Dataset creation:
   (a) Select the desired feature set.
   (b) Select the Argus client to use.
   (c) Choose whether to keep or discard Argus's management flows.
   (d) Create the dataset in the format of a CSV file.
5. Dataset labelling:
   (a) Select the path for the ground-truth file.
   (b) Generate a CSV file of the labelled dataset.

This workflow of HERA is also visually represented in Figure 3.1 with the different components properly identified as well as their relation to each requirement that will be expanded in the next section.

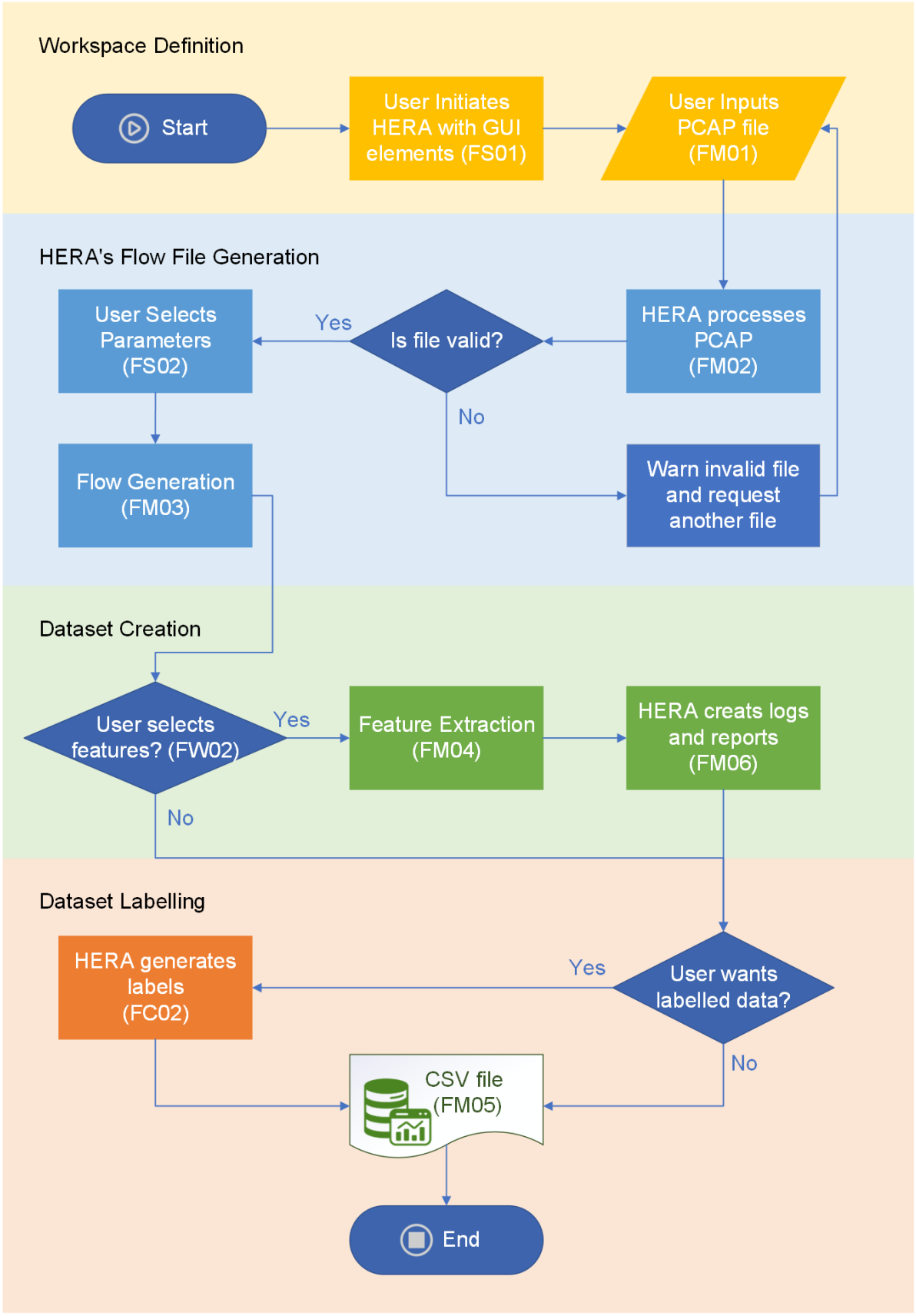


Figure 3.1: HERA's flowchart with the main components identified and the fulfilled requirements.

3.1 Requirements

The analysis of Chapter 2 allows us to establish the minimum requirements for HERA to function effectively as a traffic analysis tool for generating datasets. An examination of the trends in dataset usage, specifically the use of packet versus flow information, revealed a predominant use of flows, particularly in recent years. As illustrated in Figure 3.2a, 70% of the analysed datasets are composed of flows. This observation led to the decision to focus on flow information and, consequently, to utilize *Flow Exporters* for the type of datasets generated by HERA.

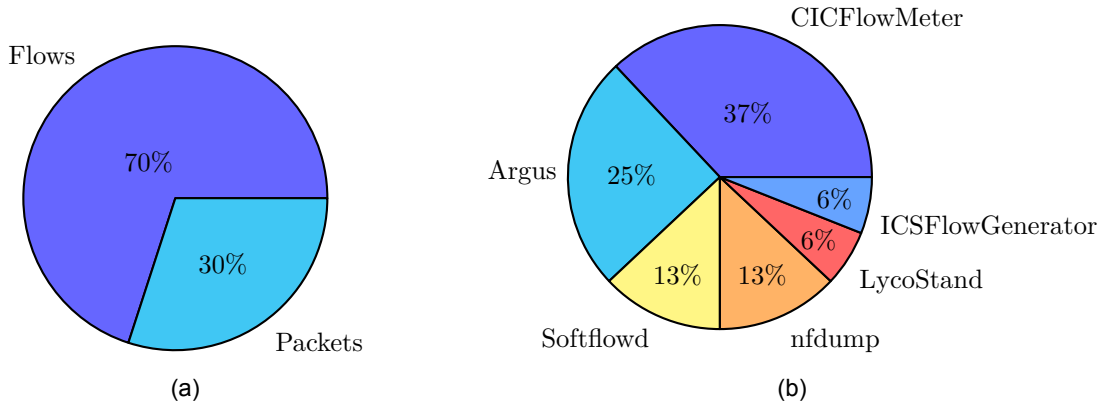


Figure 3.2: (a) Distribution of dataset types (packet or flows). (b) Distribution of flow exporters used. The data for both distributions were extracted using Table 2.4. For (b), only datasets using tools with functionalities of *Flow Exporter* as listed in Table 2.3 were considered.

Based on the findings in Section 2.2.8, two main functions were identified as essential for dataset generation: the functionalities of a *Flow Exporter* with Feature Extraction capabilities and a *Capturer*. Regarding traffic capture for the dataset, it was determined that HERA did not need to capture packet information directly but should be capable of processing PCAP files. This approach aligns with many analysed datasets, where traffic is captured and saved as a PCAP file, and subsequently, a *Flow Exporter* processes this file to create flows. Therefore, the use of *Flow Exporters* in datasets utilizing flow information was examined. As shown in Figure 3.2b, CICFlowMeter is one of the most commonly used tools. However, as discussed in Section 2.2.1.1, this tool has several issues, making it necessary to select an alternative *Flow Exporter* for HERA. In the same figure, Argus appears as the second most commonly used *Flow Exporter* with a 25% appearance in the analysed datasets. For this reason and others that will be further discussed in Section 3.2, Argus was chosen as the underlying *Flow Exporter*.

Additionally, to design an easy-to-use tool, a user interface was necessary, preferably with a Graphical User Interface (GUI) that would allow users to upload PCAP files for flow aggregation and customize parameters related to the generated flow information. For this reason, it was crucial that HERA included the following key functionalities:

- User-friendly PCAP file uploading and processing.

- Customizable parameters for flow information exportation.
- *Flow Exporter* functionalities for generating flow information and Feature Extraction.
- Intuitive GUI elements for ease of use
- Outputting the processed dataset in CSV format.

Figure 3.3 illustrates the process that HERA must use to create datasets to accomplish these objectives. It is important to note that "Network Interface" and "Capturer" are represented with dotted lines, indicating that while they are not directly integrated into the tool, their roles are crucial for providing the information needed.

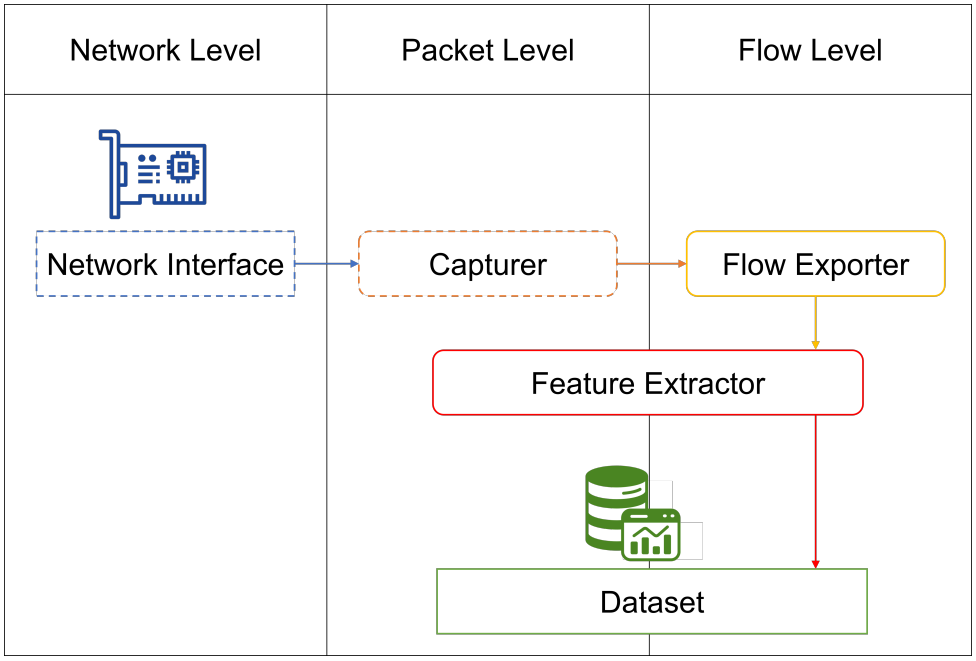


Figure 3.3: Classification of tools necessary for HERA.

After considering the overview of the required functions of HERA, both functional and non-functional requirements were established. The functional requirements were categorized into three groups: "MUST", "SHOULD", and "COULD". Firstly, the "MUST" category encompasses essential functionalities that the tool must have to be considered operational. Without these functionalities, the tool could not be considered functional. Secondly, the "SHOULD" category includes functionalities that would enhance the tool but are not necessary to meet the minimum essential functions previously outlined. Finally, the "COULD" category comprises additional features that would be beneficial but are of lower priority. These requirements are detailed in Tables 3.2, for the functional requirements, and Table 3.3, for the non-functional requirements.

It's noteworthy that most requirements were fulfilled, except FC01, FC03 and FC04 from the "COULD" category, which were considered low priority and couldn't be implemented due to time constraints.

Table 3.2: Tool's Functional Requirements.

ID	Name	Type	Description
FM01	Input	MUST	The software must accept PCAP files.
FM02	PCAP Reading	MUST	The software must be able to read network packets from captured files.
FM03	Flow Generation	MUST	The software must be able to form flows based on the input of captured files.
FM04	Feature Extraction	MUST	The software must calculate and store a predefined set of features for each flow, including identifying and classifying network protocols.
FM05	Output	MUST	The software must produce CSV files.
FM06	Logging and Reporting	MUST	The software must produce network traffic reports (flow counts, flow averages, among others.)
FS01	GUI	SHOULD	The software should have a user-friendly interface to interact with the tool.
FS02	Customization	SHOULD	The software should allow users to select features to extract and change flow definition parameters.
FC01	Anomaly Detection	COULD	The software could implement mechanisms for anomaly detection in the captured files (DDoS, and others.).
FC02	Flow Anomaly Labelling	COULD	The software could label flows as benign or malicious.
FC03	Integration	COULD	The software could support integration with other network monitoring tools.
FC04	Packet Capturing	COULD	The software could capture packets.

Table 3.3: Tool's non-functional requirements.

ID	Name	Description
NF01	Performance	The software must handle large volumes of network traffic efficiently.
NF02	Usability	The software must be easy to use with proper documentation.
NF03	Resource Utilization	The software must optimize resource usage.

3.2 Flow Exporter

This section will delineate the implementation of the flow generation process that HERA achieves with the selected *Flow Exporter*: Argus. It was designed to capture and report network flow data with capabilities for real-time network traffic analysis. The diverse set of clients within Argus enables it to perform these functions effectively, fostering seamless integration with other tools used in network environments. Argus operates on various platforms, including Linux, FreeBSD, and macOS, with minimal system requirements for deployment. The only prerequisite is the support for the libpcap library by the operating system. These versatile functionalities position Argus as a suitable choice for use within HERA, offering extensive and customizable feature sets and the ability to define fields to generate flows. Moreover, with over 30 years of contributions and continuous updates up to the present time, Argus is a reliable choice, with a scheduled release for this year, 2024, of a new version of the tool [79]. Additionally, the European Union Agency for Cybersecurity (ENISA) references Argus in some of their guides [80], to use for network forensics, mentioning it as a well-known tool for flow capturing and analysis, alongside other widely recognized tools like tcpdump, Snort, and Wireshark. Furthermore, Argus has been used in numerous projects, including dissertations, theses, and research articles[83]. These projects involve, for example, using Argus as a *Flow Exporter* to later save the results as a dataset, generating popular datasets such as UNSW-NB15 and Bot-IoT, as analysed in Section 2.

As mentioned in Section 2.2.1.6, Argus is an open-source *Flow Exporter* comprised of two modules. The first module, known as “argus-server”, performs packet processing, transforming them into network flows. The second module, “argus-client”, includes a collection of *argus* data processing programs. These programs range from allowing the extraction of flow features to the sorting of data or even allowing the use of databases. Table 3.4 lists the versions of Argus used in HERA development.

Table 3.4: Versions of the *Flow Exporter*, Argus, used to implement HERA.

Module	Version
Argus-server	3.8.0.2
Argus-client	3.8.0.2

This collection of tools provided by “argus-server” and “argus-client” can be divided into three main categories. The most commonly used Argus tools include *argus*, which is responsible for the *Flow Exporter* and *Capturer* functions, and the main clients. To note, this will be the nomenclature used in this thesis. When referring to the collection of tools, Argus will be used, but for referring to the specific tool within Argus that generates the flows, the mentioned name will be *argus*. Additionally, the main clients are part of a collection responsible for processing the flow data generated by *argus*. These clients handle tasks such as feature extraction, analysis,

[83] https://openargus.org/publications

aggregation, splitting, anonymization, and sorting of *argus* data for network flow monitoring and management.

Table 3.5 elaborates on the functions of “argus-server”, the package responsible for *argus*, and Table 3.6 details the main clients. The information regarding the function of the Argus programs was retrieved from the official website[84].

Table 3.5: Argus’s main tool.

Service	**Function**
argus	Generate flow records from PCAP files or by running as a daemon and reading packets from the network interface. This tool produces a file, commonly referred to as *argus* file or *argus* data.

Table 3.6: Argus’s clients.

Service	**Function**
ra	Reads, filters and prints *argus* data.
radium	Responsible for data collection, analytics and distribution. It is a combination of the features from *ra* and *argus* used to process a large network providing a single point of access.
racluster	Aggregates *argus* data based on the flow key criteria that can be specified by the user. It reduces the number of generated flows.
rasplit	Splits *argus* data into consecutive structured files based on size, count time or flow event. By default, it limits the resulting files to 10.000 records.
rabins	Processes *argus* data into structures based on time, input size or count. It is a combination of *rasplit* and *racluster*.
racount	Counts the number of objects in the *argus* data stream, providing a total record, source and destination packet and byte counts.
ranonymize	Strips and anonymizes fields from *argus* data.
rasort	Sorts *argus* data based on specified criteria by the user.

[84] https://www.openargus.org/documentation

Argus also provides additional programs that extend the capabilities of Argus data processing. These programs can be used by researchers to manage and analyse higher amounts of flow data, by offering functionalities such as: file conversion into argus files, event detection, address filtering, statistical analysis, flow metadata labelling based on addresses, traceroute printing, policy testing, database interactions, real-time streaming and data processing, record stripping, service validation, Address Resolution Protocol (ARP) monitoring, and time range determination. Table 3.7 lists these additional programs within Argus to help users further analyse the generated flows.

Table 3.7: Argus's additional programs that read, parse and process argus data.

Service	**Function**
raconvert	Converts ascii flow data, like CSV files or BRO-IDS/Zeek files, into *argus* files.
radump	Reads *argus* data and prints decoded user data based on tcpdump.
raevent	Reads *argus* data and prints the contents of the events found.
rafilteraddr	Provides high performance address matching by reading *argus* files and returning a list of found Internet Protocol (IP) addresses.
ragrep	Reads *argus* data and greps the result based on the specified command.
rahisto	Provides statistics based on specified criteria of *argus* data.
ralabel	Adds descriptor labels to flows in *argus* data based on addresses.
rapath	Prints traceroute path information from *argus* data.
rapolicy	Reads *argus* data and tests it against Cisco's access control list configuration file.
rasql	Reads *argus* data from a MySQL database.
rasqlinsert	Writes *argus* data into MySQL tables.
rasqltimeindex	Reads *argus* data from a MySQL database and provides fast access to the data, basing the indexes on time.
rastream	Extends the features of *rasplit*, acting as a stream block processor. *rasplit* only works for data that has fully been processed, *rastream* uses time delay to process flows as they finish arriving.
rastrip	Strips *argus* records based on specified criteria.
ratop	Displays periodically a sorted list of flow records, it is used for realtime *argus* data processing.
raservices	Discovers and validates network services, adding them to the *argus* record if the user chooses.
rarpwatch	Provides classic arpwatch functionality to *argus* data.
ratimerange	Returns the time span of *argus* data.

Finally, Argus provides a set of Perl scripts as experimental programs for functions such as printing network topology information, reporting IP address inventory and reporting application port usage. Table 3.8 presents a summary of the functions of each script.

Table 3.8: Argus perl scripts.

Service	**Function**
radark	Discovers scanner behaviour and reports it.
radecode	Uses text2pcap and tshark to decode only TCP and UDP flows from an *argus* file.
ragraph	Prints topology information derived from *argus* data.
rahosts	Reports IP address inventory from *argus* data.
raports	Reports application port usage from *argus* data.

To summarize, the programs that were selected to use in HERA were, *argus*, *ra*, *racluster* and *racount*. These Argus tools allowed the previously established objectives to be fulfilled by using them in two different HERA components. For the first one explored in this section, the *Flow Exporter* functionality, *argus* allowed to form the flows from the PCAP files that HERA receives as input. For the second one, further described in the next Section 3.3, either *ra* or *racluster* is chosen by the user to extract the features that will be present in the dataset, while *racount* is used to obtain statistics on the generated flows. If the user selects the *ra* option the dataset will be created with all generated flows alongside the selected features, however, if the user selects *racluster*, the flows will be aggregated based on the tuples that define the flows, and a smaller dataset will be returned with the same flow configuration settings. This allows some flexibility to the user to deal with larger datasets with more efficient processing with summarized information if choosing *racluster*, or to get a more detailed and individual record analysis if choosing *ra*.

Regarding the implementation of the HERA flow file generation, the process is structured into three main steps. Firstly, the code begins by allowing users to select arguments for flow generation using a “SelectMultiple” widget. This widget presents a list of available arguments that *argus* allows to use when generating flows, with a predefined set already selected by default. Users have the option to modify this selection or proceed with the default set. Table 3.9 lists the available arguments.

Secondly, HERA includes “IntText” and “Button” widgets initialized with a default value of 60. Users can adjust this value and confirm their selection using the button. A validation function ensures the entered value is positive; if negative, it prompts the user for correction. Once validated, the chosen value is stored.

Finally, users initiate flow file generation by clicking another “Button” widget. This action triggers functions that provide status messages on the file generation process. Additionally, a function utilizes the stored values to execute a subprocess call to *argus*, thereby generating the flow file.

Table 3.9: Flow generation arguments. Default set in italic.

Command	Description
-A	*Generate application byte metrics in each audit record.*
-O	Turn off Berkeley Packet Filter optimizer. If you think it generates bad code.
-Z	*Collect packet size information for all flows (to generate mean, max, min and standard deviation).*
-J	*Generate packet performance (jitter) data in each audit record.*
-m	Provide MAC address information in argus records.
-R	Generate argus records such that response times can be derived from transaction data.

3.3 Dataset Creation

The dataset creation component of HERA is organized into four sections. Initially, HERA prompts the user to perform feature selection, this is because, as mentioned in Section 2, depending on the area of study of the dataset, the necessity of certain features will vary. For this reason, the tool allows the user to select features from 130 options. This is the total number of features Argus allows to be extracted plus two, which are calculated by HERA. These two features use other preconfigured features to determine the number of connections with the same service for each source and destination address. This was implemented as Argus does not natively provide these features, and it is a useful metric as a high count of the same service being used by the same source address to different destinations might indicate scanning activity.

Alongside the list of features, HERA offers five pre-defined feature sets: all features, default features, UNSW-NB15 dataset features, BoT-IoT features, and a set of 10 features similar to the ones present in CIC-IDS2017. Regarding the default set of features, they were determined through an analysis of the most commonly present features as analysed in the state-of-the-art datasets in Section 2. These features include:

- **“FlowID”**: flow identifier.
- **“rank”**: the ordinal value of the flow record, representing its sequence number.
- **“stime”** and **“ltime”**: record start and last time.
- **“sport”** and **“dport”**: source and destination port number.
- **“saddr”** and **“daddr”**: source and destination IP address.
- **“proto”**: transaction protocol.
- **“bytes”**, **“sbytes”** and **“dbytes”**: total, source to destination and destination to source transaction bytes.

- **“pkts”**, **“spkts”** and **“dpkts”**: total, source to destination and destination to source packet count.
- **“dur”**: total duration of the flow record.
- **“runtime”** and **“idle”**: total active flow runtime and time since the last packet activity.
- **“flgs”**: flow state flags.
- **“tcpopt”**: flag for the Transmission Control Protocol (TCP) connection state.
- **“Ssaddr”** and **“Sdaddr”**: calculated features for the number of connections with the same service and source or destination address, respectively.

For the other sets, they were created to facilitate the selection of features for the user. For all features, it allows, as the name implies, to select all features. This enables the user to observe the results of all features obtained before deciding which final ones to keep that are the most relevant for their specific needs. The dataset-related features were selected based on public datasets, two that used Argus, UNSW-NB15 and BoT-IoT, and CIC-IDS2017. This allows the user to generate datasets with the same features as these for an easier comparative analysis.

Additionally, the user can choose between *ra* or *racluster* using a “Select” widget, and whether to print management records generated by *argus* (defaulting to *ra* and no printing). Next, using Python libraries listed in Table 3.10, HERA iterates over its flow files to generate CSV datasets with the selected features. It is noteworthy that core features always included are “rank”, “stime”, “ltime”, “proto”, “saddr”, “daddr”, “sport” and “dport”. Moreover, it constructs a unique “FlowID” by merging destination and source addresses, ports, and protocol separated by hyphens. Finally, HERA verifies if two calculated features were selected and generates them by counting connections with the same service for each source and destination address. With all features extracted to a CSV file, the dataset creation component concludes by invoking a subprocess of *racount* to store information such as packet counts and total number of flows in a text file.

Table 3.10: Versions of the Python libraries used in the dataset creation component.

Library	Version
pandas	1.5.3
numpy	1.24.2

3.4 Dataset Labelling

Regarding the last HERA component, it operates in two distinct stages. Initially, users must provide a CSV file corresponding to ground truth data for the captured traffic. This file contains the correct labels identifying when attacks in the network were performed. For this step, it is important that the data was collected in a controlled environment that permits the accurate description of all activity that occurred, namely attacks. For HERA to process it, this file must include headers for “StartTime”, “LastTime”, “Proto”, “SrcAddr”, “Sport”, “DstAddr”, “Dport”, and “Label”. These fields correspond to the start and end time of a flow, the protocol used, the source and destination IP address and port, and the label that categorizes the type of attack. To note, fields can be left empty as labelling is still possible even if not all fields are filled in, and researchers may not have information for every field.

In the second stage, HERA processes each unlabeled dataset file that was produced in the previous component. For each CSV file, the tool iterates through and labels flows based on the provided information of the ground truth file. To begin, HERA verifies the time range of the unlabelled dataset CSV files, to define which rows of the ground truth to compare the data with. This is because the ground truth file must contain all information about the attacks, but the unlabelled dataset can be distributed across multiple files. This ensures the labelling is efficient. Afterwards, it utilizes a labelling function to check each flow against the selected rows of the ground truth file and compares the “StartTime”, “LastTime”, “Proto”, “SrcAddr”, “Sport”, “DstAddr” and “Dport” fields. If a flow’s characteristics match the attack’s timeframe and other fields, it is labelled as corresponding to that attack; otherwise, it is labelled as benign.

Upon completion, the tool generates a text file summarizing the total number of flows categorized into each attack type and benign traffic category. This output provides researchers with an overview of traffic classification based on the provided ground truth, supporting further analysis and evaluation.

3.5 HERA GUI

This section will demonstrate the GUI of HERA achieved with the widgets functionalities of Jupyter Notebook with the “ipywidgets” (version 8.1.1) library, in each of the four components, in order from the workspace definition to the dataset labelling.

In the workspace definition component, HERA prompts the user with input boxes to define the paths for the location of the PCAP files (Figure 3.4a), the HERA’s flow files (Figure 3.4b) (whether already generated or to be generated), and the location where the unlabelled CSV files (Figure 3.4c) should be stored. HERA then validates whether the provided locations exist, prompting the user to indicate the location again if they do not exist. Furthermore, it saves the file names of the PCAP files to be used later when generating the HERA flow files, consisting of *argus* data, and CSV files.

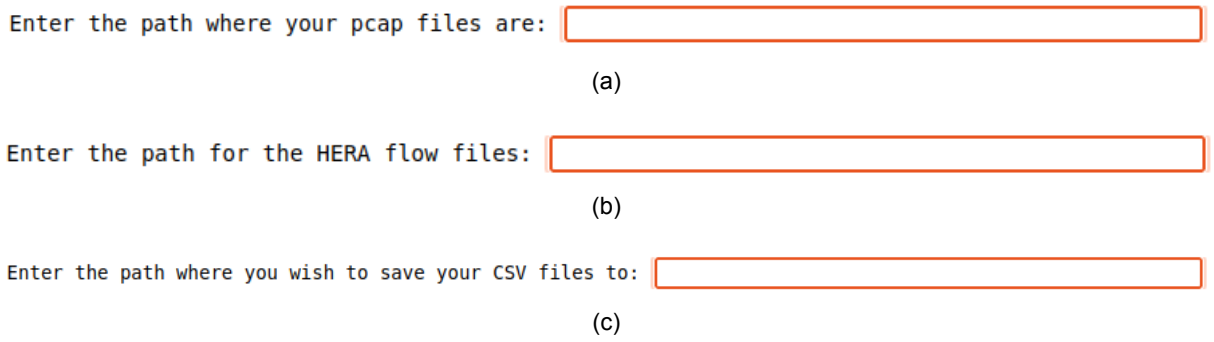


Figure 3.4: Tool's prompt requiring the user to input the path location for the PCAP (a), HERA's flow file (b) and CSV (c) files.

In the HERA's flow file generation component, which is entirely optional if the user already has them, the user is provided with options to select the arguments (Figure 3.5a) they want to use for generating the file and specify the desired flow intervals (Figure 3.5b). After selecting the options, the user can initiate the generation process by clicking a button and the flow file, which is essentially a *argus* data file, will be automatically saved in the previously selected path location. Throughout the process, the tool provides feedback to the user (Figures 3.5c, 3.5d and 3.5e), indicating whether the file generation is still in progress.

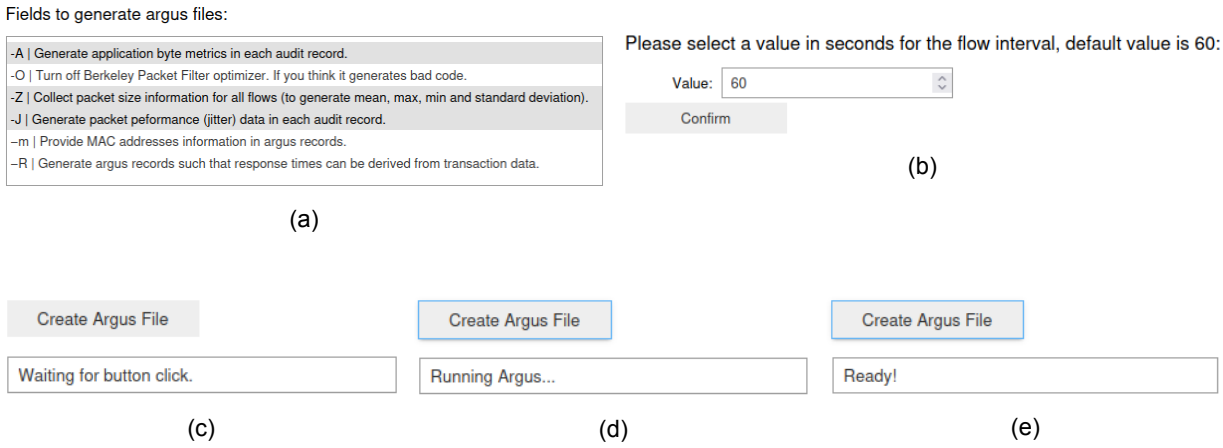


Figure 3.5: (a) Arguments to generate the HERA's flow file. (b) Selector for the flow interval, with a minimum value of 0. (c, d, e) "Create Argus File" button along with messages indicating its three stages.

The following component, dataset creation, begins with the user selecting from the 130 features to be generated. The tool provides several options in the form of buttons for feature selection: "Select all", "Default Features", "UNSW-NB15", "Bot-IoT" and "CIC 10 Features". Figure 3.6 illustrates the interface for selecting features, providing users with clear and easy-to-use options for generating their desired dataset.

This component includes two additional options. Firstly, client selection, where the user can choose which client to use for generating the dataset, as illustrated in Figure 3.7a (Table 3.6

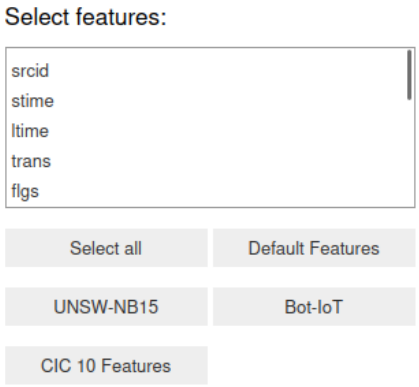
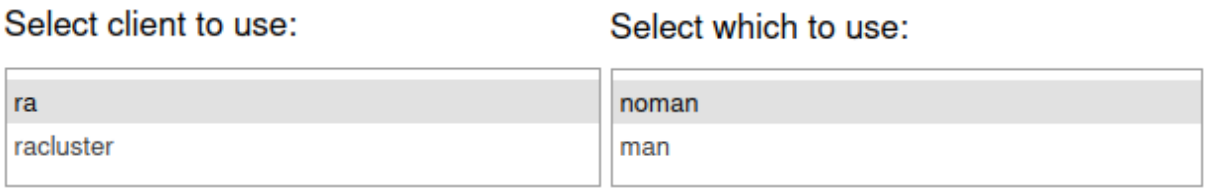


Figure 3.6: Tool's options of feature sets.

details the differences between the available clients). Secondly, management flow selection, where the user can decide whether to include or exclude Argus's management flows in the dataset, as shown in Figure 3.7b. These two options allow for further customization of the dataset based on the user's requirements.



(a) (b)

Figure 3.7: (a) Selection of which Argus's client to use. (b) Selection of whether to keep or discard management flows.

After this process, the dataset is generated according to the user's selections alongside a file containing statistics of the generated flows using *Racount*. Upon successful completion, the tool returns a message stating, "Features extracted to CSV file:" followed by the path and file name of the generated file. This message confirms that the dataset creation process has been completed and provides the location of the newly created CSV file for the user's convenience.

For the final component, dataset labelling, the tool requires the user to input the CSV file location for the ground truth file as depicted in Figure 3.8. This is the last GUI element as the next step only requires the user to run the next cell and the labelled dataset will be generated.

Figure 3.8: Tool's prompt requiring the user to input the path location for the ground truth file.

Chapter 4

Results and Analysis

For the evaluation of the tool, the files from two public datasets were used: UNSW-NB15 and CIC-IDS2017. In specific, their Packet Capture (PCAP) files were inputted on **H**olistic n**E**twork featu**R**es **A**ggregator (HERA) to generate datasets. This allowed us to compare the results obtained with the tool with those obtained by the researchers, having each served different purposes. UNSW-NB15 was used to evaluate whether the tool was utilizing Argus correctly, while CIC-IDS2017 was employed to compare the results obtained when the dataset was generated using CICFlowMeter.

4.1 UNSW-NB15

As mentioned in Section 2.3.2.1, UNSW-NB15 is a dataset used for NID research, created to address the limitations of older datasets, such as KDDCUP'99 and NSL-KDD. The researchers created the dataset data in a synthetic environment using an IXIA tool testbed for two days, 22 January and 17 February 2015, collecting 16 and 15 hours of traffic respectively. The network had 3 servers, 2 for benign traffic and one for malicious attacks, that communicated with 6 clients via 2 routers, with 3 clients each, that were separated by a firewall. This firewall was configured to accept all traffic, normal or abnormal, that was collected by tcpdump, installed in router 1. For the generation of attacks, IXIA was configured to generate one attack per second on the first day, and ten attacks per second on the second day. For the generation of the dataset, Argus and BRO-IDS/Zeek were used to create flows and features, which were saved in a database by matching the features of the two tools. Additionally, twelve algorithms were run to generate additional features. The final result was a flow-based dataset saved in Comma-Separated Values (CSV) format with normal and malicious traffic, composed of Fuzzers, Analysis attacks, Backdoors, Denial of Service (DoS), Exploits, Generic attacks, Reconnaissance, Shellcode and Worms.

The researchers also provided to the research community all files that were generated during the experiment, organizing them into five folders: "Argus Files", "BRO Files", "CSV Files",

"PCAP files" and "Reports". The "Argus Files" folder contains *argus* files generated from the PCAPs present in "PCAP files". This folder is divided into two days, containing 99.1GB of the raw traffic that was simulated. Day 22 has 53 PCAP files, while day 17 has 27 files. The division in two days can be found in every folder, using the number of PCAP files as nomenclature, except for "CSV Files" and "Reports". This means each "Argus Files" folder has 53 and 27 *argus* files that also include the respective CSV files that were created using the extracted features from the *argus* files. Similarly, the "BRO Files" folder contains BRO-IDS/Zeek log files organized by the name of the corresponding PCAP file. The "CSV Files" folder comprises the full dataset, which combines the flows of Argus and BRO-IDS/Zeek, divided into four files. Additionally, it contains documentation regarding the dataset, such as a list of events, feature sets, and ground truth. It also includes a folder with training and testing sets. Finally, the "Reports" folder provides reports of the simulations conducted on the two days.

Due to this methodology with the use of Argus and because the researchers make all their files available[85], UNSW-NB15 is a good option to compare the results obtained with HERA using their PCAP files. Moreover, because the *argus* files are included, and HERA uses these same file types, it is possible to establish a comparison between the results.

Regarding the full dataset, its four CSV files present the following time ranges in Greenwich Mean Time (GMT):

- **UNSW-NB15_1.csv**: Thursday, 22 January 2015 11:50:14 to 19:44:02.
- **UNSW-NB15_2.csv**: Thursday, 22 January 2015 19:44:02 to 00:25:24 and Wednesday, 18 February 2015 00:23:28 to 03:45:29.
- **UNSW-NB15_3.csv**: Wednesday, 18 February 2015 03:45:29 to 09:00:09.
- **UNSW-NB15_4.csv**: Wednesday, 18 February 2015 09:00:09 to 12:21:08.

It is noteworthy that the dataset was created in a different timezone than the one followed by this work. Henceforth, unless referred otherwise, the timezone used will be GMT.

Upon analysing the provided data, it was noted that only data from day 18, could be utilized to test HERA. This decision was made due to an issue with the PCAP files provided for day 22, where some PCAP files were repetitions, and for this reason, the data could not accurately be compared. In contrast, no repeated data was observed for day 18. Table 4.1 presents the division of PCAP files used from day 18 in each of the complete UNSW-NB15 dataset files. To note, day 18 traffic was primarily found in UNSW-NB15_3.csv and UNSW-NB15_4.csv, while UNSW-NB15_1.csv and UNSW-NB15_2.csv were composed of day 22 traffic with the additional files 1 to 7 from day 18. For this reason, only files 8 to 27 from day 18 were used.

[85] https://unsw-my.sharepoint.com/:f:/g/personal/z5025758_ad_unsw_edu_au/EnuQZZn3XuNBjgfcUu4DIVMBLCHyoLHqOswirpOQifr1ag?e=gKWkLS

Table 4.1: Identification of which PCAP files from day 18 belong to which dataset file.

Dataset File	PCAP Corresponding Files
UNSW-NB15_2.csv	1.pcap, 2.pcap, 3.pcap, 4.pcap, 5.pcap, 6.pcap and 7.pcap
UNSW-NB15_3.csv	7.pcap, 8.pcap, 9.pcap, 10.pcap, 11.pcap, 12.pcap, 13.pcap, 14.pcap, 15.pcap, 16.pcap, 17.pcap, 18.pcap and 19.pcap
UNSW-NB15_4.csv	19.pcap, 20.pcap, 21.pcap, 22.pcap, 23.pcap, 24.pcap, 25.pcap, 26.pcap and 27.pcap

Afterwards, the PCAPs 8 to 27 were used as input on HERA, generating the flow files and two versions of the dataset, one unlabelled and another labelled. With these files, four tests were performed. The first one consisted of comparing the HERA flow files to those provided by the researchers. This allowed us to confirm if HERA was generating the flows as expected. Then, two Machine Learning (ML) tests were performed with supervised and unsupervised learning models to understand if there were differences between the dataset generated by HERA and the one provided by the authors of UNSW-NB15, since the original dataset contained BRO-IDS/Zeek data that is not present in the HERA generated CSV files. Finally, with the labelled dataset from HERA the traffic was analysed in comparison with the original files regarding malicious flows and a model was trained to assert if HERA produced better results.

For the first test, UNSW-NB15's *argus* files and those generated by HERA were examined. During the comparison, two observations were made: first, both HERA and the provided *argus* files produce the same amount of flows. This demonstrates the implemented tool produces the correct flow generation. However, when analysing the *argus* only CSV files provided by UNSW-NB15, some differences were noted compared to those obtained with HERA, particularly in the last file. The reason for the differences between the CSV files is thought to be related to the processing of the flows and features before being turned to CSV files. This is further confirmed by two features in the official dataset being wrong, namely "swin" and "dwin". These features represent the source and destination Transmission Control Protocol (TCP) window advertisement size, values that vary widely from 0 and 255, the two values that are always generated. Furthermore, when the *argus* files that were provided are processed by Argus these values get generated equally to those in the dataset by HERA. Secondly, a significant difference was found in the total flow count between the HERA files and the final CSV files, with UNSW-NB15_3.csv having 700.001 flows and UNSW-NB15_4.csv 440.044 for a total of 1.140.045 flows. These discrepancies between the totals may be attributed to Zeek/BRO-IDS adding 339.727 flows to the dataset. Table 4.2 demonstrates the described discrepancies between the provided CSV files using exclusively Argus and the ones produced by HERA.

Table 4.2: Discrepancies in the total number of flows. The difference is calculated as the third column minus the second. First row with columns name included.

File	Number of Flows		Difference
	Researcher's CSV	HERA's CSV	
8.csv	39.312	39.312	0
9.csv	41.989	42.004	15
10.csv	38.233	38.233	0
11.csv	40.196	40.197	1
12.csv	40.712	40.729	17
13.csv	42.125	42.135	10
14.csv	36.680	36.680	0
15.csv	36.677	36.677	0
16.csv	43.818	43.833	15
17.csv	37.661	37.661	0
18.csv	34.276	34.276	0
19.csv	36.130	36.130	0
20.csv	41.511	41.524	13
21.csv	42.301	42.301	0
22.csv	43.314	50.631	7.317
23.csv	36.222	36.222	0
24.csv	40.667	40.668	1
25.csv	42.840	42.854	14
26.csv	42.846	42.858	12
27.csv	42.810	12.191	-30.619
Total	800.320	777.116	-23.204

Subsequently, ML techniques were employed to evaluate the quality of the generated datasets starting with supervised learning. Three models, Random Forest, Light Gradient Boosting Machine and Extreme Gradient Boosting [81, 82], were tested with different parameters to determine which yielded the best results. These models were chosen based on their performance and efficiency on large datasets. Feature importance analysis was conducted using Random Forest's attribute "feature_importance_" in scikit-learn[86] to select the most relevant features to train the models, resulting in: 'sbytes', 'dbytes', 'sttl', 'dttl', 'smeansz', 'dmeansz' and 'synack'. The results obtained with each algorithm are presented in Table 4.3.

The Random Forest algorithm achieved the best results and was therefore selected for training. Afterwards, this model was trained using UNSW-NB15_3.csv and UNSW-NB15_4.csv and used to classify two versions of the UNSW-NB15 dataset: one generated with HERA for files 8 to 27, named "HERA's CSV", and another consisting of the researcher's equivalent CSV files,

[86] https://scikit-learn.org/stable/

Table 4.3: Results of different algorithms applied on the UNSW-NB15 dataset.

Algorithm	Precision	Recall	F1-score	Accuracy
RandomForestClassifier	**0.98**	**0.98**	**0.98**	**0.9894**
LGBMClassifier	0.97	0.97	0.97	0.9890
XGBClassifier	0.97	0.97	0.97	0.9888

named “Researcher’s CSV”. Notably, due to the presence of NaN values in HERA’s version, an imputer using the mean strategy was employed. Because this phase was done without labelled datasets, the traditional validation on the model could not be performed, and as such the comparison focused on the proportion of malicious flows identified.

The original dataset, consisting of data from UNSW-NB15_3.csv and UNSW-NB15_4.csv files, contained 21.61% malicious traffic. Despite the model’s good performance on training, the model classified both HERA’s CSV and the researcher’s CSV, with less than half of this percentage, as depicted in Figure 4.1. Between the two, HERA’s CSV identified slightly more malicious flows. This difference likely stems from the variation in total flows, as depicted in Table 4.2.

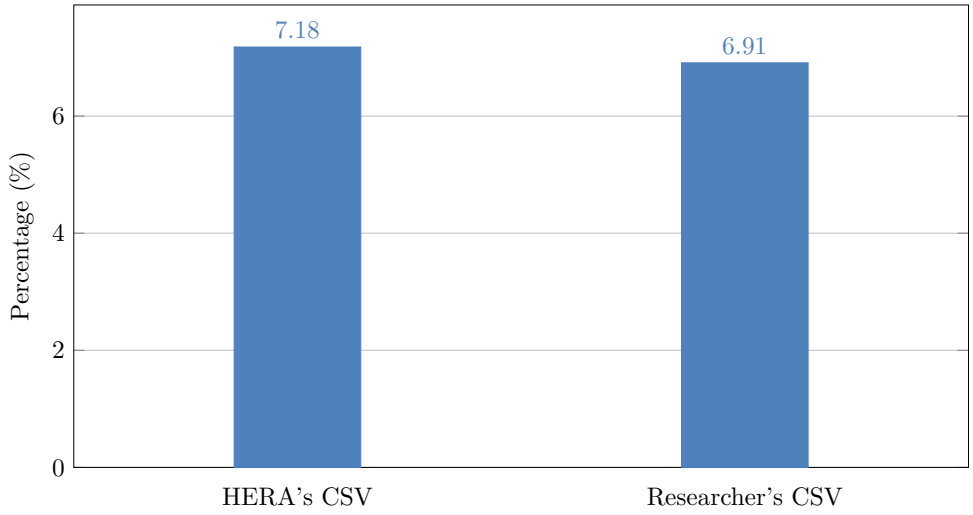


Figure 4.1: Percentage of malicious traffic in the different versions of the dataset.

For the final ML test using UNSW-NB15, an unsupervised model, K-means [83], was used with the same features as before. Using a clustering of two for the same files mentioned previously, the models obtained the following Silhouette Scores presented in Table 4.4. The Silhouette Score is a metric to evaluate the quality of a clustering algorithm’s results, ranging from -1 to 1. This metric measures how similar an object is to its cluster compared to other clusters, with a higher value indicating better-defined clusters. For this reason, the results indicate that the most well-defined clusters were obtained by HERA’s version, demonstrating an improvement using this tool.

Table 4.4: Silhouette Scores of the unsupervised mode, K-means with k=2.

Model	Silhouette Score
UNSW-NB15_3.csv + UNSW-NB15_4.csv	0.5061
HERA's CSV	**0.7258**
Researcher's CSV	0.6879

Figure 4.2 presents the results obtained with unsupervised learning concerning the percentage of identified malicious flows in the three iterations of the same PCAP files. To note, the original combination of UNSW-NB15_3.csv with the UNSW-NB15_4.csv has 21.61% malicious traffic. Similarly, the unsupervised model classified these two files with a combined 22.57% malicious traffic. Furthermore, supervised and unsupervised learning obtained similar results. For HERA's version, the supervised model classified it with 7.18%, while unsupervised classified it with 7.21%. For the Researcher's CSV the same can be observed with the supervised model obtaining 6.91% and the unsupervised 6.84%. This seemingly reinforces the idea that the difference in flows between the original UNSW-NB15 dataset and the one generated by HERA stem from BRO-IDS/Zeek adding more flows, in particular, malicious flows, to the dataset.

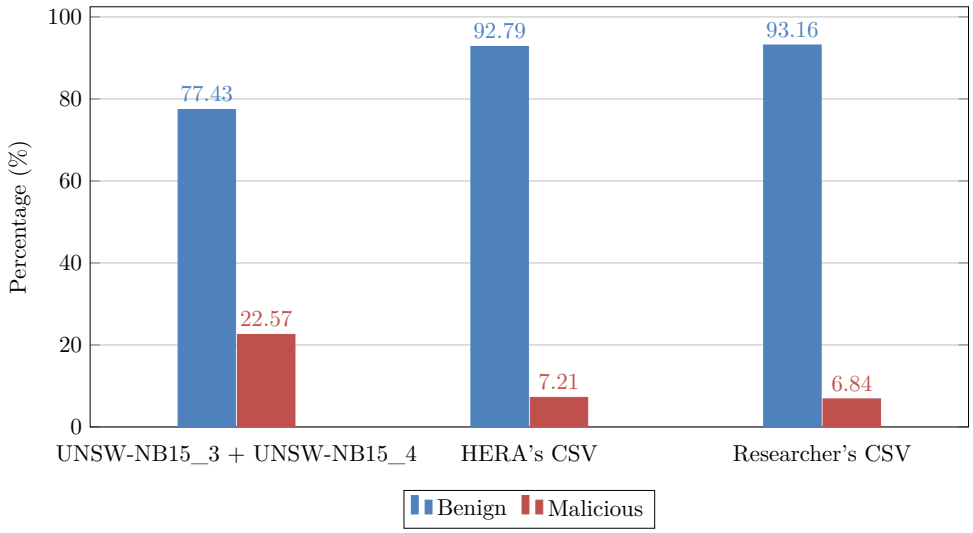


Figure 4.2: Percentage of malicious traffic in the different versions of the dataset, using k=2.

The final test, labelling the dataset with the provided ground truth file confirmed this by obtaining 6.56% of labelled flows as malicious, a close result to those that have been obtained using ML models. Table 4.5 presents the results of the labelling process by totalling the flow counts by each classification type. When comparing the total amounts of UNSW-NB15_3.csv with UNSW-NB15_4.csv, it is possible to observe the significant difference between the two datasets. With the BRO-IDS/Zeek data, UNSW-NB15 contains over 10 times the total amount of Generic attack traffic, and all attack types and benign traffic produce higher results.

Additionally, the three models, Random Forest, Light Gradient Boosting Machine and Ex-

Table 4.5: Results of the labelling process.

Traffic Classification	HERA's CSV	UNSW-NB15_3.csv + UNSW-NB15_4.csv
Benign	726.153	893.726
Exploits	16.157	28.013
Fuzzers	12.060	14.527
Generic	11.468	180.076
Reconnaissance	7.366	9.112
DoS	2.294	10.549
Shellcode	953	964
Analysis	309	1.543
Backdoor	232	1.425
Worms	104	110
Total	777.096	1.140.045

treme Gradient Boosting were used on the labelled HERA dataset obtaining the following results showcased in Table 4.6. As expected, the models weren't as successful in learning what malicious traffic was as the ones trained with UNSW-NB15_3.csv with UNSW-NB15_4.csv. This is because these news models had significantly fewer records of malicious traffic to learn from.

Table 4.6: Results of different algorithms applied on the labelled HERA dataset.

Algorithm	Precision	Recall	F1-score	Accuracy
RandomForestClassifier	**0.88**	**0.88**	**0.88**	**0.9842**
LGBMClassifier	0.88	0.87	0.87	0.9837
XGBClassifier	0.88	0.87	0.87	0.9834

4.2 CIC-IDS2017

As mentioned in Section 2.3.2.6 CIC-IDS2017 is part of the set of datasets produced by Canadian Institute for Cybersecurity (CIC) and is well-known with widespread use. However, due to the faults encountered by the tool that produced it, CICFlowMeter, its reliability is questioned.

Regarding the collected data, the researchers built the dataset by generating realistic benign background traffic of 25 users using Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS), File Transfer Protocol (FTP), Secure Shell (SSH) and email protocols. The network was divided in two, one for the victims and another for the attackers with a firewall separating them. The attack network was composed of three Windows machines and a Kali Linux, while the victim network had a Domain Name System (DNS) server, two Ubuntu web servers, and 10 machines, four Ubuntu, five Windows and one Mac from Apple. This data was generated from July 3, 2017, to July 7, 2017, totalling 5 days. For Monday (July 3), no malicious traffic was generated, while for the other days, the following attacks were performed as depicted in Table 4.7.

Table 4.7: Attacks present in CIC-IDS2017

Day	Attacks
Tuesday (July 4)	FTP and SSH Brute Force
Wednesday (July 5)	DoS (Slowloris, Slowhttptest, Hulk, GoldenEye) and Heartbleed
Thursday (July 6)	Web (morning) and Infiltration (afternoon) Attacks
Friday (July 7)	Botnet (morning), Distributed Denial of Service (DDoS) (afternoon) and PortScans (afternoon)

The researchers provide the PCAP files and the corresponding CSV files for the 5 days they produced traffic. For Monday, Tuesday, and Wednesday, a single file is provided for each day. For Thursday and Friday, the files are divided into two and three parts, respectively.

To perform the ML tests, three additional datasets, that aimed to correct the issues present in the CIC-IDS2017 dataset, as mentioned in Section 2.3.2.6, were used to train models. These corrected datasets were produced by the following authors: Engelen et al. [22], Lanvin et al. [25], and Rosay et al. [23, 24], hereafter referred to as the Engelen, Lanvin, and LYCOS-IDS2017 datasets, respectively. Notably, the dataset Lanvin [25] has multiple versions, as the authors explored different approaches to improve the dataset. The version used in this work referred to as “RP” by their authors, consists of reordered PCAP traffic with new Scan Port labels added. Additionally, using HERA, two datasets for each day were generated: one using the client *ra* and another using *racluster* to test which obtained the best result. This is because by using a different underlying *Flow Exporter*, the flows would be generated differently from CICFlowMeter and using a client (*racluster*) that helps to process bigger datasets might be beneficial. Table 4.8 presents the details of the datasets, namely the number of flows each contains.

Table 4.8: Total flow count by day for each dataset. First row with columns name included.

Dataset	Monday	Tuesday	Wednesday	Thursday	Friday	Total
CIC-IDS2017	529.919	445.910	692.704	458.969 [1]	703.248 [2]	2.830.750
Englen	371.625	322.079	496.642	362.077	547.558	2.099.981
Lanvin	372.426	322.463	497.116	355.619	548.829	2.096.453
LYCOS-IDS2017	367.182	319.638	495.238	112.482	542.963	1.837.503
HERA-Ra	436.428	378.873	565.745	411.403	599.107	2.391.556
HERA-Racluster	249.517	211.993	226.970	260.502	397.931	1.346.913

[1] Day divided into 2 files, morning and afternoon with 170.367 and 288.602 respectively. Thursday morning has 458.968 lines, but only until 170.367 does it have data.

[2] Day divided into 3 files, morning, afternoon with DDoS and afternoon with Port Scan. They have 191.034, 225.746 and 286.468 respectively.

For the first round of tests, a similar process to that used for UNSW-NB15 was employed, using the PCAP files for Wednesday. This day was selected to train models for each dataset, CIC-IDS2017, Englen, Lanvin and LYCOS-IDS107, and then used to identify malicious traffic in the datasets generated by HERA using *ra* and *racluster*. The following CICFlowMeter features were utilised for training the models: “Fwd IAT Mean’, “Flow IAT Max”, “Flow IAT Min”, “Bwd IAT Mean”, “Bwd IAT Max”, “Bwd IAT Min”, “Active Mean”, “Active Std”, “Active Max” and “Active Min”. It is important to note that HERA does not have the same features as CICFlowMeter, and they are calculated differently. Therefore, the closest equivalent features in HERA were selected for use in the models to identify malicious traffic. These equivalents are as follows: “SIntPkt”, “SIntPktMax”, “SIntPktMin”, “DIntPkt”, “DIntPktMax”, “DIntPktMin”, “Mean”, “StdDev”, “Max” and “Min”.

Table 4.9 demonstrates the results of the best iteration that each algorithm produced, organized by dataset, having in all versions the algorithm eXtreme Gradient Boosting (XGBoost) perform the best and was therefore applied to all datasets. As previously, due to the lack of labels, the percentage of benign and malicious traffic was used to compare the results. Due to corrections made to the CIC-IDS2017 dataset, the corrected versions contain fewer flows, resulting in a slight variation in the identified percentage of malicious traffic. However, this percentage remains around 35 to 36%. In specific, the percentages of malicious traffic were 36.48% for CIC-IDS2017, 35.74% for Englen, 34.56% for Lanvin and 35.6% for LYCOS-IDS2017. However, when the models were applied to the HERA-Ra and HERA-Racluster datasets, similar results to the UNSW-NB15 dataset were observed, with a lower percentage of flows detected as malicious compared to the original datasets. Noteworthy exceptions include the model using CIC-IDS2017, which showed higher rates of identified malicious traffic. This is believed to be due to overfitting, where benign traffic was incorrectly identified as malicious. Another difference was observed with the LYCOS-IDS2017 dataset. Unlike its counterparts, LYCOS-IDS2017 identified more malicious flows when using HERA-Ra and fewer when using

HERA-Racluster, a phenomenon opposite to that seen with the other datasets. Figure 4.3 illustrates the different percentages of malicious traffic obtained by the models compared to the original datasets on which the models were trained.

Table 4.9: Results of different algorithms applied on the datasets CIC-IDS2017, Englen, Lanvin and LYCOS-IDS2017.

Algorithm	Precision	Recall	F1-score	Accuracy
CIC-IDS2017				
RandomForestClassifier	0.98	0.73	0.84	0.8970
LGBMClassifier	0.90	0.96	0.93	0.9451
XGBClassifier	**0.90**	**0.97**	**0.93**	**0.9492**
Englen				
RandomForestClassifier	0.99	0.99	0.99	0.9933
LGBMClassifier	0.99	0.99	0.99	0.9946
XGBClassifier	**1.00**	**1.00**	**1.00**	**0.9974**
Lanvin				
RandomForestClassifier	0.99	0.99	0.99	0.9939
LGBMClassifier	0.99	0.99	0.99	0.9950
XGBClassifier	**1.00**	**1.00**	**1.00**	**0.9969**
LYCOS-IDS2017				
RandomForestClassifier	1.00	0.99	0.99	0.9958
LGBMClassifier	0.99	0.99	0.99	0.9953
XGBClassifier	**1.00**	**0.99**	**0.99**	**0.9961**

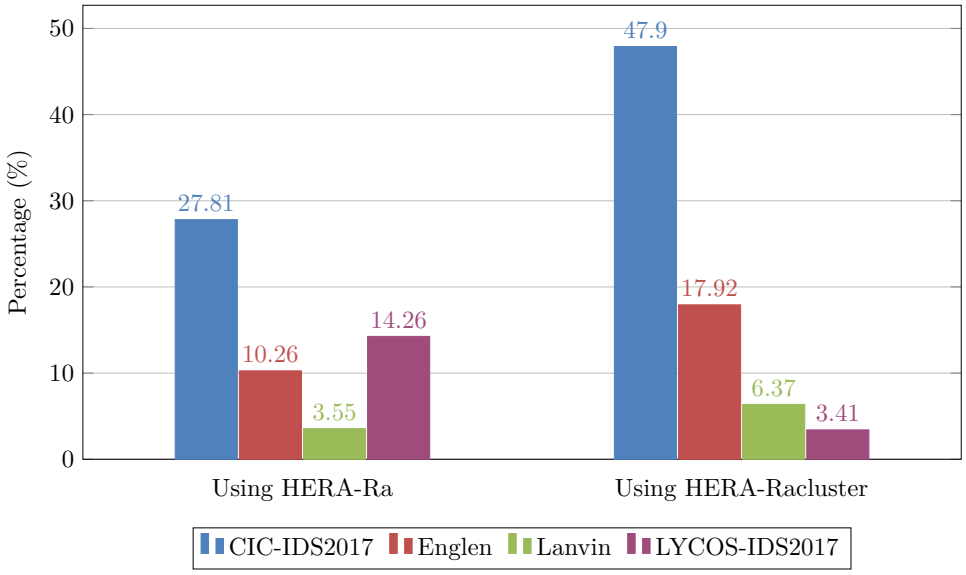


Figure 4.3: Percentage of malicious traffic in the two versions produced by HERA obtained by four different models.

Another test involved using K-means clustering for unsupervised anomaly detection on the three versions of the CIC-IDS2017 dataset. For this test, the traffic from Monday and Tuesday was combined for both the CIC-IDS2017 dataset and the datasets generated by HERA, using both *ra* and *racluster*. This generated the following flow totals:

- **CIC-IDS2017** had a total of 975.827 flows.
- **HERA-Ra** had a total of 635.488 flows.
- **HERA-Racluster** had a total of 451.762 flows.

From these datasets, the same features from the supervised models were used, with NaN values being removed. Two tests were conducted with 2 and 3 clusters. To evaluate the performance of the models, a Silhouette Score was calculated, yielding the following results presented in Table 4.10

Table 4.10: Silhouette Scores of the unsupervised model.

Model	Silhouette Score
HERA-Ra, k=2	0.7815
HERA-Ra, k=3	0.8152
HERA-Racluster, k=2	0.8003
HERA-Racluster, k=3	0.8009
CIC-IDS2017, k=2	0.9200
CIC-IDS2017, k=3	**0.9262**

Regarding the results obtained, the percentage of malicious traffic identified using unsupervised learning on the CIC-IDS2017 dataset remained approximately the same as the real percentage. However, for the HERA-Ra and HERA-Racluster datasets, the percentage of identified malicious traffic was significantly higher. This discrepancy may be due to Argus's method of generating flows, where benign flows might be grouped more frequently compared to CICFlowMeter, thereby increasing the overall proportion of malicious traffic. Figures 4.4 and 4.5 illustrate the percentage of malicious traffic present in the original CIC-IDS2017 dataset and the datasets used for unsupervised learning, with k=2 and k=3 clusters, respectively. To note, the real amount of malicious traffic present in CIC-IDS2017 for the combination of Monday and Tuesday is 98.58% benign with 1.42% malicious, of the second 0.81% and 0.6% is for FTP and SSH Brute Force respectively.

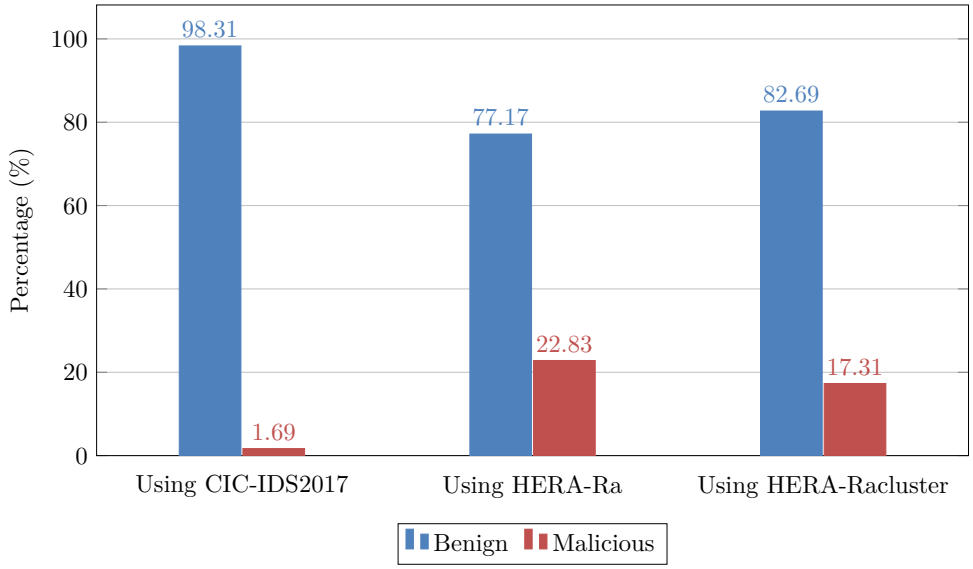


Figure 4.4: Percentage of malicious traffic in the datasets, using k=2.

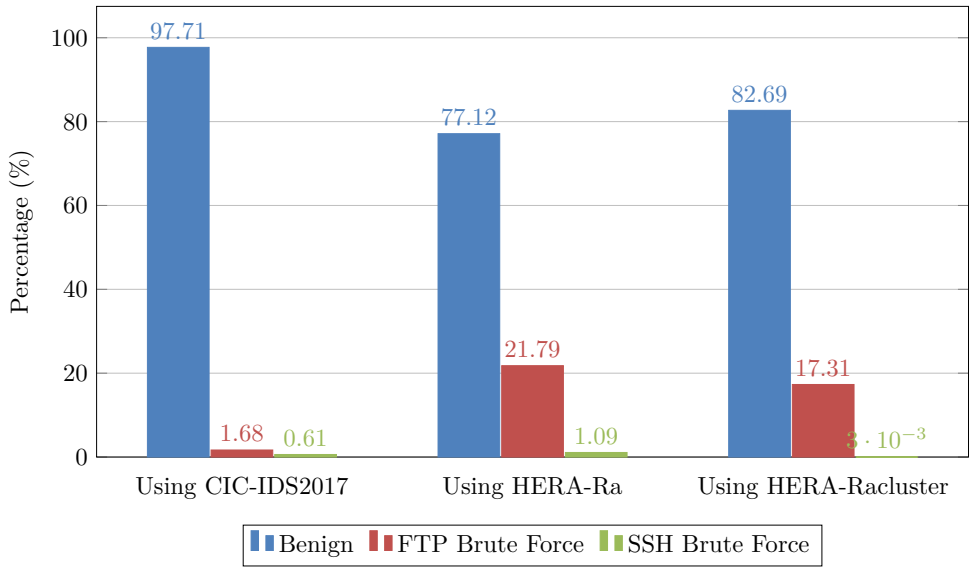


Figure 4.5: Percentage of malicious traffic in the datasets, using k=3.

Finally, using HERA, the traffic from Monday + Tuesday and Wednesday were labelled. To note, unlike UNSW-NB15, the researchers do not provide the ground truth file, but provide the times in which attacks were performed alongside the Internet Protocol (IP) addresses used in their website[87]. However when the times provided were compared to the times present in the labelled CIC-IDS2017 dataset differences were found as present in Table 4.11. Only for the DoS-GoldenEye and Heartbleed attacks, no differences were found. To ensure a good comparison, the times from the CIC-IDS2017 CSV were used to create the ground truth file to label the HERA dataset.

[87] https://www.unb.ca/cic/datasets/ids-2017.html

Table 4.11: Timeranges of the CIC-IDS2017 attacks.

Attack	Described Timeranges	CIC-IDS2017 Timeranges
FTP Brute Force	July 4, 09:20 – 10:20	July 4, 09:17 - 10:30
SSH Brute Force	July 4, 14:00 – 15:00	July 4, 14:09 - 15:11
DoS-Slowloris	July 5, 09:47 – 10:10	July 5, 09:01 – 10:11
DoS-Slowhttptest	July 5, 10:14 – 10:35	July 5, 10:15 – 10:37
DoS-Hulk	July 5, 10:43 – 11:00	July 5, 10:43 – 11:07
DoS-GoldenEye	July 5, 11:10 – 11:23	July 5, 11:10 – 11:23
Heartbleed	July 5, 15:12 - 15:32	July 5, 15:12 - 15:32

Regarding the obtained results, Table 4.12 present the total flow amounts for the two datasets, the one generated using HERA and CIC-IDS2017 for the Monday + Tuesday traffic, while Figure 4.6 present the percentage of the benign and malicious traffic.

Table 4.12: Results of the labelling process for Monday and Tuesday traffic.

Traffic Classification	HERA's CSV	CIC-IDS2017
Benign	808.337	961.992
FTP Brute Force	4.003	7.938
SSH Brute Force	2.959	5.897
Total	815.299	975.827

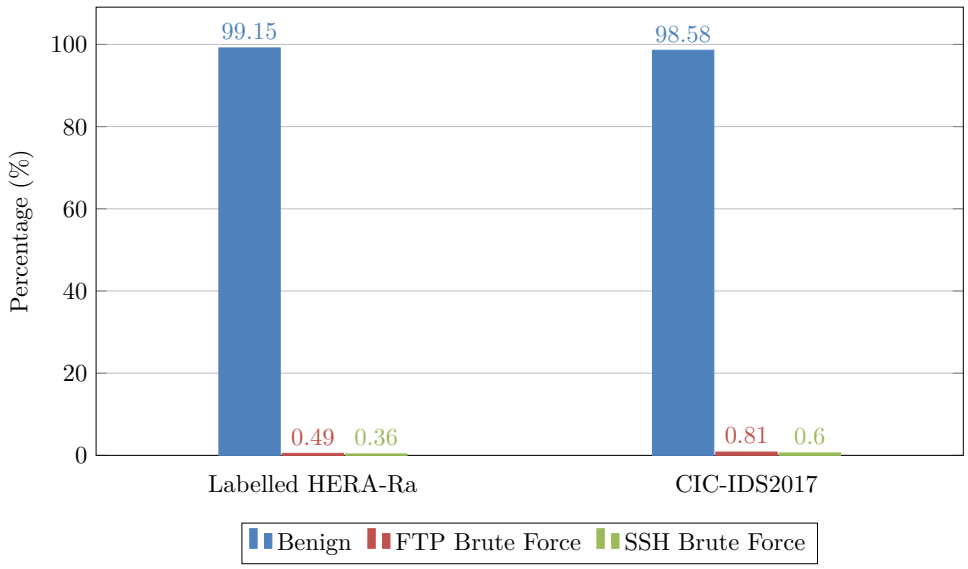


Figure 4.6: Percentage of traffic types in the datasets for Monday + Tuesday traffic.

Similarly, for the Wednesday traffic, Table 4.13 presents the total amount of flows and Figure 4.7 showcases the percentages of the traffic types.

Table 4.13: Results of the labelling process for Wednesday traffic.

Traffic Classification	HERA's CSV	CIC-IDS2017
Benign	377.804	440.031
DoS-Hulk	161.225	231.073
DoS-Slowhttptest	8.995	5.499
DoS-GoldenEye	8.972	10.293
DoS-Slowloris	8.729	5.796
Heartbleed	19	11
Total	565.744	692.703

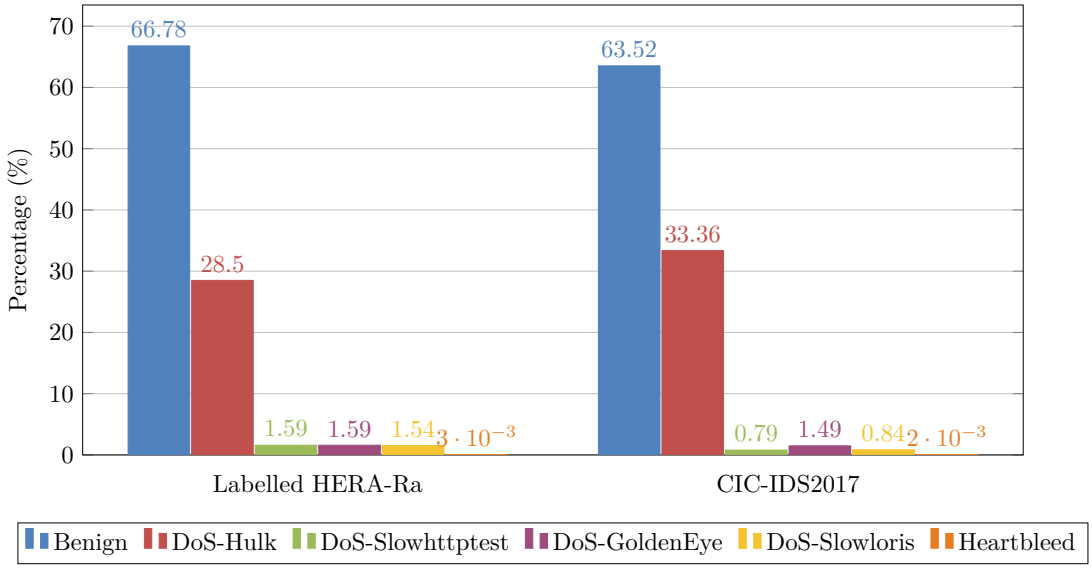


Figure 4.7: Percentage of traffic types in the datasets for Wednesday traffic.

This demonstrates that HERA yielded similar percentages of traffic to CIC-IDS2017, even if fewer flows are exported. Furthermore, ML models were trained using the Wednesday traffic, which results are present in Table 4.14. This dataset using HERA obtained similar results to the alternatives presented to correct CIC-IDS2017 as stated in the literature review [22–25], presenting the highest values for accuracy of all models.

Table 4.14: Results of different algorithms applied on the labelled HERA dataset using Wednesday traffic.

Algorithm	Precision	Recall	F1-score	Accuracy
RandomForestClassifier	**1.00**	**1.00**	**1.00**	**0.9975**
LGBMClassifier	1.00	1.00	1.00	0.9974
XGBClassifier	**1.00**	**1.00**	**1.00**	**0.9975**

Chapter 5

Conclusions and Future work

This chapter summarizes the main conclusions of this thesis, offering a recapitulation of the results obtained during the state-of-the-art analysis and outlining the achieved objectives. Additionally, the limitations of the proposed solution and potential improvements are discussed.

5.1 Research Summary & Main Findings

This thesis explores current practices and methods in dataset creation for use in Network Intrusion Detection (NID) research. It begins with an analysis of the tools used in Network Traffic Analysis (NTA), which generate the data for these datasets, and then examines popular datasets and their features to provide a clear understanding of the current standards being applied. Lastly, the thesis presents a new open-source solution to dataset creation to address current known issues with popular tools. In this regard, all initially proposed objectives were accomplished, with the main results for each objective being:

- **Objective 01**: A literature review of the state-of-the-art was performed on NTA tools analysing their functions and categorizing them based on their most common uses in the dataset creation process. Additionally, the review established a timeline of the appearance of NID datasets, analysed their characteristics, and examined the most commonly used features, grouping them based on their representations. This analysis uncovered several issues within the dataset creation process, including faults in the tools used to obtain data, the processes used to create the datasets, and the lack of standardization in dataset features. Additionally, from this systematic review, a paper titled “A Review on Intrusion Detection Datasets: Tools, Processes, and Features” was created that is currently under review.
- **Objective 02**: An open-source NTA tool, **H**olistic n**E**twork featu**R**es **A**ggregator (HERA), was developed using Jupyter Notebook’s widgets to enhance the user experience. This tool, leveraging the underlying flow exporter Argus, generates network flows with para-

meters that can be easily customized. Furthermore, it allows users to select from a wide array of features for inclusion in the final dataset. Finally, by the user providing a ground truth file, HERA returns a labelled dataset. In summary, this tool facilitates quick and easy dataset creation, which can be utilized in NID research.

- **Objective 03**: Two types of tests were performed to validate the tool's execution. First, to verify the correct generation of flows, the UNSW-NB15 dataset was used. The tool successfully generated the same number of flows in the **argus** file as those produced in the original file by the researchers. Second, Machine Learning (ML) algorithms were applied to the UNSW-NB15 and CIC-IDS2017 datasets to perform anomaly detection. The results showed that the dataset created with the implemented tool produced similar results to those obtained using the original Comma-Separated Values (CSV) files of **argus** information provided by the authors, demonstrating the tool's effectiveness. For the CIC-IDS2017 dataset, results varied widely, highlighting the differences between the two tools in generating flows.

5.2 Limitations & Future Work

Despite the successful implementation of the tool, there are limitations and opportunities for future work. While the most important requirements were fulfilled, some envisioned features could not be implemented due to time constraints, and others could potentially be achieved in different ways.

Firstly, regarding certain requirements that were not met:

- **FC01**: Anomaly detection of the produced datasets was not implemented. This feature would have been a beneficial addition to the process of dataset labelling. It would allow us to classify traffic into binary categories of benign and malicious, providing a good foundation for subsequent analysis of the dataset using ML.
- **FC03**: Integration with other tools was not realized. This could have enabled real-time analysis and utilization of Argus's own real-time functionalities.
- **FC04**: Packet capturing was not implemented. While this is not necessary since the tool uses already saved data, which also ensures reproducibility, adding this feature would allow the tool to handle the entire dataset creation process, without needing the functions of an extra *Capturer* tool.

Secondly, a requirement that could be improved:

- **FS01**: Although the Graphical User Interface (GUI) requirement was met using Jupyter Notebook widgets, developing a standalone tool with a conventional GUI would be beneficial. This would provide users the option to use a typical tool without needing Jupyter Notebook, while still allowing the Jupyter Notebook version for streamlined ML analysis.

Bibliography

[1] Nour Alqudah and Qussai Yaseen. Machine learning for traffic analysis: A review. *Procedia Computer Science*, 170:911–916, 2020. ISSN: 1877-0509. The 11th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops. doi:10.1016/j.procs.2020.03.111.

[2] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón. UGR'16: A new dataset for the evaluation of cyclostationarity-based network IDSs. *Computers and Security*, 73:411–424, 2018. ISSN: 01674048 (ISSN). Elsevier Ltd. doi:10.1016/j.cose.2017.11.004.

[3] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013. ISSN: 1084-8045. doi:https://doi.org/10.1016/j.jnca.2012.09.004.

[4] Suman Lata and Dheerendra Singh. Intrusion detection system in cloud environment: Literature survey & future research directions. *International Journal of Information Management Data Insights*, 2(2):100134, 2022. ISSN: 2667-0968. doi:https://doi.org/10.1016/j.jjimei.2022.100134.

[5] Ankit Thakkar and Ritika Lohiya. A review of the advancement in intrusion detection datasets. *Procedia Computer Science*, 167:636–645, 2020. ISSN: 1877-0509. International Conference on Computational Intelligence and Data Science. doi:10.1016/j.procs.2020.03.330.

[6] Luis Dias and Miguel Correia. *Big Data Analytics for Intrusion Detection In P. Ganapathi & D. Shanmugapriya (Eds.), Handbook of Research on Machine and Deep Learning Applications for Cyber Security*, chapter 14, pages 292–316. IGI Global, 01 2020. doi:10.4018/978-1-5225-9611-0.ch014.

[7] T. Sowmya and E.A. Mary Anita. A comprehensive review of ai based intrusion detection system. *Measurement: Sensors*, 28:100827, 2023. ISSN: 2665-9174. doi:10.1016/j.measen.2023.100827.

[8] T. Saranya, S. Sridevi, C. Deisy, Tran Duc Chung, and M.K.A.Ahamed Khan. Performance analysis of machine learning algorithms in intrusion detection system: A review. *Procedia Computer Science*, 171:1251–1260, 2020. ISSN: 1877-0509. Third International Conference on Computing and Network Communications (CoCoNet'19). doi:https://doi.org/10.1016/j.procs.2020.04.133.

[9] Lisa Liu, Gints Engelen, Timothy Lynar, Daryl Essam, and Wouter Joosen. Error prevalence in NIDS datasets: A case study on CIC-IDS-2017 and CSE-CIC-IDS-2018. In *2022 IEEE Conference on Communications and Network Security (CNS)*, pages 254–262. IEEE, 2022. doi:10.1109/CNS56114.2022.9947235.

[10] Mohanad Sarhan, Siamak Layeghy, and Marius Portmann. Evaluating standard feature sets towards increased generalisability and explainability of ML-based network intrusion detection. *Big Data Research*, 30:100359, 2022. ISSN: 22145796. Elsevier. doi:10.1016/j.bdr.2022.100359.

[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. doi:10.1109/ICISSEC.2016.7885840.

[12] Matthew J Page, Joanne E McKenzie, Patrick M Bossuyt, Isabelle Boutron, Tammy C Hoffmann, Cynthia D Mulrow, Larissa Shamseer, Jennifer M Tetzlaff, Elie A Akl, Sue E Brennan, Roger Chou, Julie Glanville, Jeremy M Grimshaw, Asbjørn Hróbjartsson, Manoj M Lalu, Tianjing Li, Elizabeth W Loder, Evan Mayo-Wilson, Steve McDonald, Luke A McGuinness, Lesley A Stewart, James Thomas, Andrea C Tricco, Vivian A Welch, Penny Whiting, and David Moher. The prisma 2020 statement: an updated guideline for reporting systematic reviews. *BMJ*, 372, 2021. doi:10.1136/bmj.n71.

[13] Catrin Sohrabi, Thomas Franchi, Ginimol Mathew, Ahmed Kerwan, Maria Nicola, Michelle Griffin, Maliha Agha, and Riaz Agha. Prisma 2020 statement: What's new and the importance of reporting guidelines. *International Journal of Surgery*, 88:105918, 2021. ISSN: 1743-9191. doi:https://doi.org/10.1016/j.ijsu.2021.105918.

[14] Cisco UCS CLI System Monitoring Guide for Cisco UCS Mini, Release 3.0. https://www.cisco.com/c/en/us/td/docs/unified_computing/ucs/sw/cli/config/guide/3-0/b_UCSM_CLI_System_Monitoring_Guide_30/b_UCSM_CLI_System_Monitoring_Guide_30_chapter_01011.html, 2015. Accessed: Dec. 08, 2023.

[15] NetFlow Version 9 Flow-Record Format. https://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.html, 2011. Accessed: Jun. 22, 2024.

[16] Gernot Vormayr, Joachim Fabini, and Tanja Zseby. Why are my flows different? a tutorial on flow exporters. *IEEE Communications Surveys & Tutorials*, 22(3):2064–2103, 2020. doi:10.1109/COMST.2020.2989695.

[17] M. Swarnkar, R. Kumar, R. Baidyo, and G. Hariharasudhan. LiteEx: A lightweight feature extraction tool for captured network traces. In *Int. Conf. COMmun. Syst. NETworkS, COMSNETS*, pages 243–251. Institute of Electrical and Electronics Engineers Inc., 2023. ISBN: 978-166547706-2 (ISBN). doi:10.1109/COMSNETS56262.2023.10041389.

[18] Boryau Hsupeng, Kun-Wei Lee, Te-En Wei, and Shih-Hao Wang. Explainable malware detection using predefined network flow. In *2022 24th International Conference on Advanced Communication Technology (ICACT)*, pages 27–33. IEEE, 2022. ISSN: 1738-9445. doi:10.23919/ICACT53585.2022.9728897.

[19] Y. Li, T. Qin, Y. Huang, J. Lan, Z. Liang, and T. Geng. HDFEF: A hierarchical and dynamic feature extraction framework for intrusion detection systems. *Computers and Security*, 121, 2022. ISSN: 01674048 (ISSN). Elsevier Ltd. doi:10.1016/j.cose.2022.102842.

[20] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Mamun, and Ali Ghorbani. *Characterization of Encrypted and VPN Traffic Using Time-Related Features*. SciTePress, 2016. The International Conference on Information Systems Security and Privacy (ICISSP). doi:10.5220/0005740704070414.

[21] A.H. Lashkari, G.D. Gil, M.S.I. Mamun, and A.A. Ghorbani. Characterization of tor traffic using time based features. In Mori P., Furnell S., and Camp O., editors, *ICISSP - Proc. Int. Conf. Inf. Syst. Secur. Priv.*, pages 253–262. SciTePress, 2017. ISBN: 978-989758209-7 (ISBN). doi:10.5220/0006105602530262.

[22] Gints Engelen, Vera Rimmer, and Wouter Joosen. Troubleshooting an intrusion detection dataset: the cicids2017 case study. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 7–12. IEEE, 2021. doi:10.1109/SPW53761.2021.00009.

[23] A. Rosay, F. Carlier, E. Cheval, and P. Leroux. From CIC-IDS2017 to LYCOS-IDS2017: A corrected dataset for better performance. In *ACM Int. Conf. Proc. Ser.*, pages 570–575. Association for Computing Machinery, 2021. ISBN: 978-145039115-3 (ISBN). doi:10.1145/3486622.3493973.

[24] Arnaud Rosay, Eloise Cheval, Florent Carlier, and Pascal Leroux. Network intrusion detection: A comprehensive analysis of CIC-IDS2017. In P. Mori, G. Lenzini, and S. Furnell, editors, *Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP)*, pages 25–36. SciTePress, 2022. ISBN: 978-989-758-553-1. doi:10.5220/0010774000003120.

[25] M. Lanvin, P.-F. Gimenez, Y. Han, F. Majorczyk, L. Mé, and E. Totel. Errors in the CICIDS2017 dataset and the significant differences in detection performances it makes. In Kallel S., Jmaiel M., Hadj Kacem A., Zulkernine M., Cuppens F., and Cuppens N., editors, *Lect. Notes Comput. Sci.*, volume 13857 LNCS, pages 18–33. Springer Science and Business Media Deutschland GmbH, 2023. ISBN: 03029743 (ISSN); 978-303131107-9 (ISBN). doi:10.1007/978-3-031-31108-6_2.

[26] Diego Fernandez, Hugo Lorenzo, Francisco J. Novoa, Fidel Cacheda, and Victor Carneiro. Tools for managing network traffic flows: A comparative analysis. In *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, pages 1–5. IEEE, 2017. doi:10.1109/NCA.2017.8171373.

[27] A. Dehlaghi-Ghadim, M.H. Moghadam, A. Balador, and H. Hansson. Anomaly detection dataset for industrial control systems. *IEEE Access*, 11:107982–107996, 2023. ISSN: 21693536 (ISSN). Institute of Electrical and Electronics Engineers Inc. doi:10.1109/ACCESS.2023.3320928.

[28] Nfdump Exporter. https://github.com/phaag/nfexporter, 2021. Accessed: Jun. 22, 2024.

[29] A. Campazas-Vega, I.S. Crespo-Martínez, A.M. Guerrero-Higueras, and C. Fernández-Llamas. Flow-data gathering using netflow sensors for fitting malicious-traffic detection models. *Sensors (Switzerland)*, 20(24):1–13, 2020. ISSN: 14248220 (ISSN). Publisher: MDPI AG. doi:10.3390/s20247294.

[30] Joel Sommers, Hyungsuk Kim, and Paul Barford. Harpoon: a flow-level traffic generator for router and network tests. *ACM SIGMETRICS Performance Evaluation Review*, 32(1): 392, 2004. ISSN: 0163-5999. doi:10.1145/1012888.1005733.

[31] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15):3531–3547, 2012. ISSN: 13891286. Publiser: Elsevier Ltd. doi:10.1016/j.comnet.2012.02.019.

[32] Zied Aouini and Adrian Pekar. NFStream: A flexible network data analysis framework. *Computer Networks*, 204:108719, 2022. ISSN: 1389-1286. Elsevier Ltd. doi:10.1016/j.comnet.2021.108719.

[33] Stefan Burschka and Benoît Dupasquier. Tranalyzer: Versatile high performance network traffic analyser. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016. doi:10.1109/SSCI.2016.7849909.

[34] R. Zhao, Y. Huang, X. Deng, Z. Xue, J. Li, Z. Huang, and Y. Wang. Flow transformer: A novel anonymity network traffic classifier with attention mechanism. In *Proc. - Int. Conf. Mobil., Sens. Netw., MSN*, pages 223–230. Institute of Electrical and Electronics Engineers Inc., 2021. ISBN: 978-166540668-0 (ISBN). doi:10.1109/MSN53354.2021.00045.

[35] Y.A. Farrukh, I. Khan, S. Wali, D. Bierbrauer, J.A. Pavlik, and N.D. Bastian. Payload-byte: A tool for extracting and labeling packet capture files of modern network intrusion detection datasets. In *Proc. - IEEE/ACM Int. Conf. Big Data Comput., Appl. Technol., BDCAT*, pages 58–67. Institute of Electrical and Electronics Engineers Inc., 2022. ISBN: 978-166546090-3 (ISBN). doi:10.1109/BDCAT56447.2022.00015.

[36] 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 and Security*, 99, 2020. ISSN: 01674048 (ISSN). Elsevier Ltd. doi:10.1016/j.cose.2020.102022.

[37] K. Wilailux and S. Ngamsuriyaroj. Novel bi-directional flow-based traffic generation framework for ids evaluation and exploratory data analysis. *Journal of Information Processing*, 29:256–265, 2021. ISSN: 03875806 (ISSN). Information Processing Society of Japan. doi:10.2197/IPSJJIP.29.256.

[38] A. Ferriyan, A.H. Thamrin, K. Takeda, and J. Murai. Generating network intrusion detection dataset based on real and encrypted synthetic attack traffic. *Applied Sciences (Switzerland)*, 11(17), 2021. ISSN: 20763417 (ISSN). MDPI. doi:10.3390/app11177868.

[39] R. Damasevicius, A. Venckauskas, S. Grigaliunas, J. Toldinas, N. Morkevicius, T. Aleliunas, and P. Smuikys. Litnet-2020: An annotated real-world network flow dataset for network intrusion detection. *Electronics (Switzerland)*, 9(5), 2020. ISSN: 20799292 (ISSN). MDPI AG. doi:10.3390/electronics9050800.

[40] D. Ageyev, T. Radivilova, O. Bondarenko, and O. Mohammed. Data sets selection for distributed infocommunication networks traffic abnormality detection. In *2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology, PIC S and T 2021 - Proceedings*, pages 635–638. Institute of Electrical and Electronics Engineers Inc., 2021. ISBN: 978-166540682-6 (ISBN). doi:10.1109/PICST54195.2021.9772239.

[41] Ahmed Alshaibi, Mustafa Al-Ani, Abeer Al-Azzawi, Anton Konev, and Alexander Shelupanov. The comparison of cybersecurity datasets. *Data*, 7:22, 01 2022. doi:10.3390/data7020022.

[42] 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*, 3(4):262–294, 2000. ISSN: 1094-9224. ACM. doi:10.1145/382912.382923.

[43] S.J. Stolfo, Wei Fan, Wenke Lee, A. Prodromidis, and P.K. Chan. Cost-based modeling for fraud and intrusion detection: results from the JAM project. In *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, volume 2, pages 130–144 vol.2. IEEE, 2000. doi:10.1109/DISCEX.2000.821515.

[44] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6. IEEE, 2009. ISSN: 2329-6275. doi:10.1109/CISDA.2009.5356528.

[45] Anna Sperotto, Ramin Sadre, Frank van Vliet, and Aiko Pras. A labeled data set for flow-based intrusion detection. In Giorgio Nunzi, Caterina Scoglio, and Xing Li, editors, *IP Operations and Management*, Lecture Notes in Computer Science, pages 39–50. Springer, 2009. ISBN: 978-3-642-04968-2. doi:10.1007/978-3-642-04968-2_4.

[46] 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*, BADGERS '11, pages 29–36. Association for Computing Machinery, 2011. ISBN: 978-1-4503-0768-0. doi:10.1145/1978672.1978676.

[47] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. MAWILab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proceedings of the 6th International COnference*, Co-NEXT '10, pages 1–12. Association for Computing Machinery, 2010. ISBN: 978-1-4503-0448-1. doi:10.1145/1921168.1921179.

[48] 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. ISSN: 0167-4048. Elsevier Ltd. doi:10.1016/j.cose.2011.12.012.

[49] Elaheh Biglar Beigi, Hossein Hadian Jazi, Natalia Stakhanova, and Ali A. Ghorbani. Towards effective feature selection in machine learning-based botnet detection approaches. In *2014 IEEE Conference on Communications and Network Security*, pages 247–255, 2014. doi:10.1109/CNS.2014.6997492.

[50] 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. doi:10.1109/MilCIS.2015.7348942.

[51] 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. ISSN: 1553-877X. in IEEE Communications Surveys & Tutorials. doi:10.1109/COMST.2015.2402161.

[52] Efstratios Chatzoglou, Georgios Kambourakis, and Constantinos Kolias. Empirical evaluation of attacks against IEEE 802.11 enterprise networks: The AWID3 dataset. *IEEE Access*, 9:34188–34205, 2021. ISSN: 2169-3536. in IEEE Access. doi:10.1109/ACCESS.2021.3061609.

[53] M Ring, S Wunderlich, D Grüdl, D Landes, and A Hotho. Creation of flow-based data sets for intrusion detection. *Journal of Information Warfare*, 16(4):41–54, 2017. ISSN: 1445-3312. Peregrine Technical Solutions.

[54] Markus Ring, Sarah Wunderlich, Dominik Grüdl, D. Landes, and A. Hotho. Flow-based benchmark data sets for intrusion detection. In *European Conference on Cyber Warfare and Security (ECCWS)*, 2017. in Volume: Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS).

[55] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying IoT devices in smart

environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, 2019. ISSN: 1558-0660. IEEE Transactions on Mobile Computing. doi:10.1109/TMC.2018.2866249.

[56] I. Sharafaldin, A.H. Lashkari, and A.A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Mori P., Furnell S., and Camp O., editors, *ICISSP - Proc. Int. Conf. Inf. Syst. Secur. Priv.*, pages 108–116. SciTePress, 2018. ISBN: 978-989758282-0 (ISBN). doi:10.5220/0006639801080116.

[57] I. Sharafaldin, A.H. Lashkari, S. Hakak, and A.A. Ghorbani. Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In *Proc. Int. Carnahan Conf. Secur. Technol.*, volume 2019-October. Institute of Electrical and Electronics Engineers Inc., 2019. ISBN: 10716572 (ISSN); 978-172811576-4 (ISBN). doi:10.1109/CCST.2019.8888419.

[58] Ranjit Panigrahi and Samarjeet Borah. A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems. *International Journal of Engineering & Technology*, 7:479–482, 2018. IJET. doi:https://doi.org/10.14419/ijet.v7i3.24.22797.

[59] H.J. Hadi, U. Hayat, N. Musthaq, F.B. Hussain, and Y. Cao. Developing realistic distributed denial of service (DDoS) dataset for machine learning-based intrusion detection system. In Lloret Mauri J., Boubchir L., Jararweh Y., Benkhelifa E., and Saleh I., editors, *2022 9th International Conference on Internet of Things, Systems, Management and Security, IOTSMS 2022*. Institute of Electrical and Electronics Engineers Inc., 2022. ISBN: 979-835032045-9 (ISBN). doi:10.1109/IOTSMS58070.2022.10062034.

[60] G. Amponis, P. Radoglou-Grammatikis, G. Nakas, S. Goudos, V. Argyriou, T. Lagkas, and P. Sarigiannidis. 5g core PFCP intrusion detection dataset. In *2023 12th International Conference on Modern Circuits and Systems Technologies, MOCAST 2023 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., 2023. ISBN: 979-835032107-4 (ISBN). doi:10.1109/MOCAST57943.2023.10176693.

[61] Muhammad Ali, Mansoor Ulhaq, Hanif Durad, Anila Usman, Syed Muhammad Mohsin, Hana Mujlid, and Carsten Maple. Effective network intrusion detection using stacking-based ensemble approach. *International Journal of Information Security*, 22:1–18, 2023. Springer. doi:10.1007/s10207-023-00718-7.

[62] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset. *Future Generation Computer Systems*, 100:779–796, 2019. ISSN: 0167-739X. Elsevier. doi:10.1016/j.future.2019.05.041.

[63] Maede Zolanvari, Marcio A. Teixeira, Lav Gupta, Khaled M. Khan, and Raj Jain. Machine learning-based network vulnerability analysis of industrial internet of things. *IEEE Internet of Things Journal*, 6(4):6822–6834, 2019. ISSN: 2327-4662. IEEE Internet of Things Journal. doi:10.1109/JIOT.2019.2912022.

[64] Alejandro Guerra-Manzanares, Jorge Medina-Galindo, Hayretdin Bahsi, and Sven Nõmm. MedBIoT: Generation of an IoT botnet dataset in a medium-sized IoT network. In *6th International Conference on Information Systems Security and Privacy, ICISSP 2020*. SciTePress, 2020. doi:10.5220/0009187802070218.

[65] Sebastian Garcia, Agustin Parmisano, and Maria Jose Erquiaga. IoT-23: A labeled dataset with malicious and benign IoT network traffic, 2021. doi:10.5281/zenodo.4743746.

[66] Nour Moustafa. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_iot datasets. *Sustainable Cities and Society*, 72:102994, 2021. ISSN: 2210-6707. Elsevier Ltd. doi:10.1016/j.scs.2021.102994.

[67] M. Komisarek, M. Pawlicki, M.-E. Mihailescu, D. Mihai, M. Carabas, R. Kozik, and M. Choras. A novel, refined dataset for real-time network intrusion detection. In *ACM International Conference Proceeding Series*. Association for Computing Machinery, 2022. ISBN: 978-145039670-7 (ISBN). doi:10.1145/3538969.3544486.

[68] Muna Al-Hawawreh, Elena Sitnikova, and Neda Aboutorab. X-IIoTID: A connectivity-agnostic and device-agnostic intrusion data set for industrial internet of things. *IEEE Internet of Things Journal*, 9(5):3962–3977, 2022. ISSN: 2327-4662. doi:10.1109/JIOT.2021.3102056.

[69] Mohamed Amine Ferrag, Othmane Friha, Djallel Hamouda, Leandros Maglaras, and Helge Janicke. Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning. *IEEE Access*, 10:40281–40306, 2022. ISSN: 2169-3536. doi:10.1109/ACCESS.2022.3165809.

[70] S.S. Bagui, D. Mink, S.C. Bagui, T. Ghosh, R. Plenkers, T. McElroy, S. Dulaney, and S. Shabanali. Introducing UWF-ZeekData22: A comprehensive network traffic dataset based on the MITRE ATT&CK framework. *Data*, 8(1), 2023. ISSN: 23065729 (ISSN). MDPI. doi:10.3390/data8010018.

[71] MITRE ATT&CK. https://attack.mitre.org/, 2024. Accessed: Jun. 22, 2024.

[72] P. Gogoi, M.H. Bhuyan, D.K. Bhattacharyya, and J.K. Kalita. Packet and flow based network intrusion dataset. In *Commun. Comput. Info. Sci.*, volume 306 CCIS, pages 322–334. Springer, 2012. ISBN: 18650929 (ISSN); 978-364232128-3 (ISBN). doi:10.1007/978-3-642-32129-0_34.

[73] J. Uramova, P. Scgeč, M. Moravčík, J. Papan, M. Kontšek, and J. Hrabovský. Infrastructure for generating new IDS dataset. In Jakab F., editor, *ICETA - IEEE Int. Conf. Emerg. eLearning Technol. Appl., Proc.*, pages 603–610. Institute of Electrical and Electronics Engineers Inc., 2018. ISBN: 978-153867914-2 (ISBN). doi:10.1109/ICETA.2018.8572201.

[74] M. Nivaashini and P. Thangaraj. A framework of novel feature set extraction based intrusion detection system for internet of things using hybrid machine learning algorithms. In *Int. Conf. Comput., Power Commun. Technol., GUCON*, pages 44–49. Institute

of Electrical and Electronics Engineers Inc., 2019. ISBN: 978-153864491-1 (ISBN). doi:10.1109/GUCON.2018.8674952.

[75] A. Kaan Sarica and P. Angin. A novel SDN dataset for intrusion detection in IoT networks. In Zincir-Heywood N., Ulema M., Sayit M., Clayman S., Kim M.-S., and Cetinkaya C., editors, *Int. Conf. Netw. Serv. Manag., CNSM, Int. Workshop Anal. Serv. Appl. Manag., AnServApp Int. Workshop Future Evol. Internet Protoc., IPFuture*. Institute of Electrical and Electronics Engineers Inc., 2020. ISBN: 978-390317631-7 (ISBN). doi:10.23919/CNSM50824.2020.9269042.

[76] Basheer Husham Ali, Nasri Sulaiman, S.A.R. Al-Haddad, Rodziah Atan, and Siti Lailatul Mohd Hassan. DDoS detection using active and idle features of revised CICFlowMeter and statistical approaches. In *2022 4th International Conference on Advanced Science and Engineering (ICOASE)*, pages 148–153. IEEE, 2022. doi:10.1109/ICOASE56293.2022.10075591.

[77] Sriram Muralidharan, Susmithaa A, Vignesh B, and Dr V. End-to-end machine learning pipeline for real-time network traffic classification and monitoring in android automotive. *International Journal of Innovative Technology and Exploring Engineering*, 11:32–38, 2022. BEIESP. doi:10.35940/ijitee.G9982.0611722.

[78] M. Sarhan, S. Layeghy, and M. Portmann. Towards a standard feature set for network intrusion detection system datasets. *Mobile Networks and Applications*, 27(1):357–370, 2022. ISSN: 1383469X (ISSN). Springer. doi:10.1007/s11036-021-01843-0.

[79] Argus v5.0.0 release. https://pairlist1.pair.net/pipermail/argus/2024-June/011459.html, 2024. Accessed: Jun. 23, 2024.

[80] Introduction to Network Forensics. https://www.enisa.europa.eu/topics/training-and-exercises/trainings-for-cybersecurity-specialists/online-training-material/documents/introduction-to-network-forensics-handbook.pdf, 2019. Accessed: Jun. 23, 2024.

[81] Hongyu Liu and Bo Lang. Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences*, 9(20), 2019. ISSN: 2076-3417. doi:10.3390/app9204396.

[82] Manjula C. Belavagi and Balachandra Muniyal. Performance evaluation of supervised machine learning algorithms for intrusion detection. *Procedia Computer Science*, 89:117–123, 2016. ISSN: 1877-0509. Twelfth International Conference on Communication Networks, ICCN 2016, August 19– 21, 2016, Bangalore, India Twelfth International Conference on Data Mining and Warehousing, ICDMW 2016, August 19-21, 2016, Bangalore, India Twelfth International Conference on Image and Signal Processing, ICISP 2016, August 19-21, 2016, Bangalore, India. doi:https://doi.org/10.1016/j.procs.2016.06.016.

[83] Radi Khaoula and Moughit Mohamed. Improving intrusion detection using pca and k-means clustering algorithm. In *2022 9th International Conference on*

Wireless Networks and Mobile Communications (WINCOM), pages 1–5, 2022. doi:10.1109/WINCOM55661.2022.9966426.