



**FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE**

Threat Detection in Network Traffic using Time Series Analysis

by

Josef Koumar

A dissertation thesis submitted to
the Faculty of Information Technology, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Doctor.

Doctoral study programme: Informatics
Department of Digital Design

Prague, December 2025

Supervisor:

Assoc. prof. Tomáš Čejka, Ph.D.
Department of Digital Design
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Copyright © 2025 Josef Koumar

Abstract and contributions

This dissertation thesis deals with the advancement of network threat detection methodologies through the application of systematic time series analysis. As modern network security faces a critical visibility gap due to the near-universal adoption of encryption, traditional payload-based inspection has become largely ineffective. This research addresses this challenge by defining a taxonomy for network-derived time series to model behavioral and temporal dependencies inherent in encrypted traffic. By treating network metadata as temporal signals, the dissertation thesis develops novel detection mechanisms for identifying malicious activity, ranging from botnet Command and Control (C&C) channels to sophisticated data exfiltration and Distributed Denial of Service (DDoS) attacks, without requiring access to unencrypted payloads.

In particular, the main contributions of the dissertation thesis are as follows:

1. Formalization of a taxonomy for network time series that enables the granular analysis of temporal patterns for precise threat characterization.
2. Development of threat detection methodologies for encrypted traffic by analyzing the *Single flow time series*
3. Design of specialized periodicity mining algorithms for *Flow time series* to expose automated heartbeat patterns and periodic behaviors characteristic of malware and botnet communications.
4. Application of unsupervised forecasting-based methods for outlier detection, targeting network intrusions that deviate from established historical traffic baselines.
5. Proposed integration of Large Language Models (LLM) to automate the explanation of detected anomalies to get insights for human analysts.

Keywords: Threat Detection, Network Security, Encrypted Traffic Analysis, Anomaly Detection, Time Series, Time Series Analysis, Network Monitoring

Abstrakt

Tato disertační práce se věnuje pokroku v metodikách detekce síťových hrozeb prostřednictvím aplikace systematické analýzy časových řad. Vzhledem k tomu, že moderní síťová bezpečnost čelí kritickému nedostatku vzhledu do dat v důsledku šifrování téměř veškeré komunikace, tradiční inspekce založená na obsahu paketů se stala v podstatě nepoužitelnou. Výzkum v této disertační práci na tuto výzvu reaguje definováním taxonomie časových řad ze síťového provozu pro modelování behaviorálních a časových závislostí v šifrovaném provozu. Tato práce vyvíjí inovativní detekční mechanismy pro identifikaci škodlivých aktivit, od Command and Control (C&C) komunikace botnetů až po sofistikovanou exfiltraci dat a útoky typu Distributed Denial of Service (DDoS), a to vše bez nutnosti přístupu k nešifrovanému obsahu.

Konkrétně jsou hlavními přínosy této disertační práce:

1. Formalizace taxonomie časových řad ze síťového provozu, která umožňuje granulární analýzu časových vzorců pro přesnou charakterizaci hrozeb.
2. Vývoj metodik detekce hrozeb v šifrovaném provozu pomocí analýzy časových řad uvnitř jednotlivých toků.
3. Návrh specializovaných algoritmů pro dolování periodicity v časových řadách z více toků za účelem odhalení automatizovaných vzorců typu heartbeat a periodického chování charakteristického pro malware a komunikaci botnetů.
4. Aplikace metod založených na předpovídání pro detekci odlehlých hodnot, zaměřená na síťové hrozby, které se odchyľují od historického chování.
5. Návrh integrace velkých jazykových modelů (Large Language Models (LLM)) pro automatizovanou interpretaci detekovaných anomálií s cílem poskytnout srozumitelné vysvětlení abnormality pro lidské analytiky.

Klíčová slova: Detekce hrozeb, Bezpečnost sítí, Analýza šifrovaného provozu, Detekce anomálií, Časové řady, Analýza časových řad, Monitorování sítě

Acknowledgements

First of all, I would like to express my gratitude to my dissertation thesis supervisor, Assoc. Prof. Tomáš Čejka, Ph.D. He has been a constant source of encouragement and insight throughout my research, providing me with numerous solutions to problems and professional advancements.

My sincere thanks go to Prof. Hana Kubátová, CSc. Her constant support has been instrumental in my research, conference travels, and collaboration with foreign researchers. Beyond the technical guidance, I am truly thankful for the unique perspective she shared on navigating the realities of a career in academia.

My deepest gratitude goes to my colleague and friend Karel Hynek, Ph.D., for teaching me how to write an academic paper and conduct quality research. He made a significant contribution to publishing the results of my research. Without him, I would never have completed the PhD studies with such successful publications.

I am profoundly thankful to Kamil Jeřábek, Ph.D. As both a colleague and a friend, he has been a constant source of inspiration and new ideas. I truly appreciate his patience and his gift for turning even my most misguided ideas into something meaningful through his thoughtful critiques and constant encouragement.

I thank my friend and colleague Jaroslav Pešek for helping me think like a scientist and teaching me how to travel with joy. Also, I would like to thank my friend and colleague Richard Plný deeply for being the nicest person in the office, who made me laugh many times, especially when he had a dolphin stuffed animal or placed online bets.

I thank my colleagues for their assistance in numerous instances, both with research and personal decisions. Special thanks go to Dominik Soukup, Pavel Šiška, Václav Bartoš, Martin Žádník, Jiří Setinský, Ondřej Sedláček, and Jan Luxemburk. Moreover, I also want to thank my Bachelor's and Master's students who helped me implement some proof-of-concept ideas. Special thanks go to Timotej Smoleň, David Kežlínek, Milan Kureš, Karel Mudruňka, Dominik Oškera, Klára Nosková, Lukáš Jančíčka, and Maxim Kalvoda.

Special thanks go to the staff of the Department of Digital Design at FIT CTU in Prague and the Department of Administration and Security Tools at CESNET, who both maintained a pleasant and flexible environment for my research. I would like to express

my special thanks to the management of both for providing the funding for my research.

My research has also been partially supported by the Ministry of Education, Youth, and Sport of the Czech Republic under research programs LM2023054 and CZ.02.01.01/00/24_030/0015041, by the European Union under the research programs 833418 and 101128073, by the Ministry of the Interior under the research programs VJ02010024 and VK02030197, by the Ministry of the Defense under the research program OYCESNET20221, by the Technology Agency of the Czech Republic under the research program TH04010073, and by the Grant Agency of the Czech Technical University in Prague under the grants No. SGS20/210/OHK3/3T/18 and No. SGS23/207/OHK3/3T/18.

Finally, I would like to extend my greatest thanks to my family members and my girlfriend for their infinite patience and care. Additionally, I would like to extend my greatest thanks to my friends for the time we spent together in the pub. A memorable time in 2025 was one week's stay on a yacht, where we only drank, which significantly helped me manage my PhD studies. And no, I am not an alcoholic. Here, I provide the recipe for the "Utopit komouše" drink:

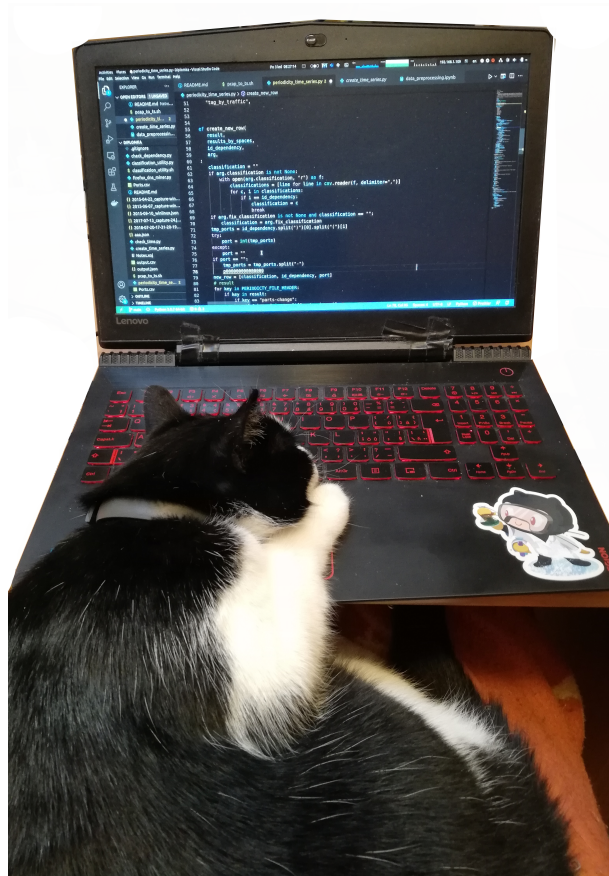
Definition 0.0.1 (Utopit komouše). The recipe for this famous drink is as follows:

1. Add vodka
2. Add griotka (english: cherry liqueur)
3. Shaken, not stirred

I recommend drinking one shot in the morning and one shot before sleep to increase your academic productivity.

Dedication

To Mňouk for writing this dissertation thesis



Contents

List of Theorems	xiii
Acronyms	xiv
1 Introduction	1
1.1 Motivation	2
1.2 Problem Statement	2
1.3 Goals of the Dissertation Thesis	3
1.4 Overview of the Dissertation Thesis	3
1.5 Structure of the Dissertation Thesis	4
1.6 Author's Publications Used in the Dissertation Thesis and His Contribution	5
1.7 Ethics Statement	8
2 Background and State-of-the-Art	9
2.1 Time Series Analysis	9
2.2 Network Traffic Monitoring	13
2.3 Network Traffic Classification	17
2.4 Network Traffic Forecasting	21
2.5 Anomaly Detection in Network Traffic	23
3 Time Series from Network Traffic	25
3.1 Types of Time Series from Network Traffic	27
3.2 Comparison of Time Series from Network Traffic	30
4 Threat Detection using Single Flow Time Series Analysis	33
4.1 Motivation	33
4.2 Exploration of features based on Time Series Analysis	35
4.3 NetTiSA flow	43
4.4 Key Findings and Contributions	53

5	Threat Detection using Flow Time Series Analysis	55
5.1	Motivation	55
5.2	Periodic Behaviors	56
5.3	Detection of Periodic Behaviors	57
5.4	Classification of Applications, Services, and Operating Systems	61
5.5	Enhanced Detection of Periodic Behaviors	62
5.6	Detection of Cryptomining	65
5.7	Detection of Botnet's C&C Communication	71
5.8	Deployment	79
5.9	Key Findings and Contributions	79
6	Threat Detection using Aggregated Time Series Analysis	81
6.1	Motivation	81
6.2	CESNET-TimeSeries24 Dataset	83
6.3	Anomalies in Network Traffic	86
6.4	Network Traffic Forecasting using SARIMA	88
6.5	Anomaly Detection System	90
6.6	Explainable Anomaly Detection	96
6.7	Key Findings and Contributions	104
7	Conclusions	107
7.1	Summary	107
7.2	Contributions of the Dissertation Thesis	108
7.3	Future Work	109
	Bibliography	111
	Publications of the Thesis Author	135

List of Figures

1.1	Overview of the dissertation	4
2.1	Example of the monitoring infrastructure	16
3.1	Comparison of evenly and unevenly spaced time series	26
3.2	Examples of <i>Packet time series</i> with cyclic behaviors	28
3.3	Example of <i>Flow time series</i>	29
3.4	Example of <i>Aggregated time series</i>	31
3.5	Relationship between types of time series from network traffic.	32
4.1	The proportion of long flows in selected datasets	34
4.2	Examples of <i>Single flow time series</i> of different traffic types.	35
4.3	Average feature importance of the presented feature set	42
4.4	Average feature importance of each feature category	42
4.5	Comparison of binary classification	45
4.6	Comparison of multiclass classification	47
4.7	Comparison of telemetry size	49
4.8	Measuring setup	50
4.9	Results of performance and memory requirements comparison	51
5.1	Periodic behavior of a time series of the malware Mirai	57
5.2	Periodic behavior of a time series of the social network Facebook	57
5.3	Example of <i>Flow time series</i>	58
5.4	Lomb-Scargle periodogram applied on <i>Flow time series</i>	58
5.5	Autocorrelation function of the <i>Flow time series</i>	59
5.6	Histograms of distances between peaks of the autocorrelation functions	60
5.7	Methodology of periodicity detection approach	60
5.8	The confusion matrix for social networks	63
5.9	The confusion matrix for services and protocols	63
5.10	Methodology of enhanced periodicity detection approach	64

5.11	Statistical evaluation of the CESNET-MINER22 dataset	67
5.12	The number of flows in <i>Flow time series</i>	68
5.13	Feature importance	69
5.14	Results of the Machine Learning (ML) pipeline	70
5.15	Capturing process of the CESNET-CC25 dataset	73
5.16	Distribution of packet lengths	75
5.17	Methodology of botnet detection	76
5.18	Experimental evaluation of detecting C&C communication	77
5.19	Deployment into high-speed Internet Service Provider (ISP) network	79
6.1	Architecture of dataset collection	83
6.2	Aggregation process for capturing <i>Aggregated time series</i>	84
6.3	Examples of the anomaly types	87
6.4	Analysis of detected anomaly	89
6.5	Example of using SARIMA model for network traffic forecasting	89
6.6	Design of the Network Outlier Detection System (NODS) architecture.	91
6.7	The evolution of detected outliers for each model.	94
6.8	Analysis of detected outliers per each IP address	95
6.9	Comparison of the number of incidents and critical incidents.	96
6.10	Evaluation methodology of explainable anomaly detection	98

List of Tables

4.1	Description of <i>Single flow time series</i> datasets	38
4.2	Final results (in %) of binary classification	39
4.3	Final results (in %) of multiclass classification	40
4.4	Comparison of full feature set and best-performing reduced feature set	43
4.5	Comparison of average results of binary and multiclass classification	47
4.6	Performance comparison on real traffic	52
4.7	The comparison of memory usage	53
5.1	Results of machine learning algorithms in %.	62
5.2	Summary of best results	69
5.3	Botnet captures inside CESNET-CC25 dataset	74
5.4	Standardized train and test split of IP flows in the dataset.	74
5.5	Comparison of botnet detection methods on the CTU-13 dataset	78
6.1	Open source software used in the NODS system.	91
6.2	The number of outliers per time series metric	96
6.3	Results of Case 1: LLM as Autonomous Anomaly Analyst Agent	101
6.4	Results of Case 2: LLM as Autonomous Anomaly Alerter Agent	102

List of Theorems

0.0.1 Definition (Utopit komouše)	vi
2.1.1 Definition (Stochastic process)	9
2.1.2 Definition (Time series)	9
2.1.3 Definition (Evenly sampled time series)	10
2.1.4 Definition (Unevenly spaced time series)	11
3.0.1 Definition (Network dependency)	25
3.1.1 Definition (Packet time series)	27
3.1.2 Definition (Single flow time series)	28
3.1.3 Definition (Flow time series)	29
3.1.4 Definition (Aggregated time series)	30
6.3.1 Definition (Anomaly)	86
6.3.2 Definition (Global anomaly)	87
6.3.3 Definition (Contextual anomaly)	87
6.3.4 Definition (Seasonal anomaly)	87
6.3.5 Definition (Subsequence anomaly)	87
6.3.6 Definition (Pattern anomaly)	87

Acronyms

R^2 Score Coefficient of determination.

ACF Autocorrelation Function.

API Application Programming Interface.

APT Advanced Persistent Threat.

AR Autoregressive.

ARIMA Autoregressive Integrated Moving Average.

C&C Command and Control.

CDF Cumulative Distribution Function.

CNN Convolutional Neural Network.

DDoS Distributed Denial of Service.

DES Double Exponential Smoothing.

DL Deep Learning.

DNS Domain Name System.

DOD Differential Outlier Detector.

DoH DNS over HTTPS.

DoS Denial of Service.

DoT DNS over TLS.

DPDK Data Plane Development Kit.

DPI Deep Packet Inspection.

ECH Encrypted Client Hello.

EDR Endpoint Detection and Response.

FFT Fast Fourier Transform.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

ICMP Internet Control Message Protocol.

IDS Intrusion Detection System.

IoT Internet of Things.

IPFIX Internet Protocol Flow Information Export.

IPS Intrusion Prevention System.

ISO/OSI International Organization for Standardization/Open Systems Interconnection.

ISP Internet Service Provider.

k-NN K-Nearest Neighbors.

LLM Large Language Models.

LOF Local Outlier Factor.

LSTM Long short-term memory.

MA Moving Average.

ML Machine Learning.

MOD Mean Outlier Detector.

MTU Maximum Transmission Unit.

NAT Network Address Translation.

Neural ODE Neural Ordinary Differential Equation.

NLP Natural Language Processing.

NN Neural Network.

NODS Network Outlier Detection System.

P2P Peer-to-Peer.

PCA Principal Component Analysis.

PoS Proof-of-Stake.

PoW Proof-of-Work.

QoS Quality of Service.

QUIC Quick UDP Internet Connections.

RMSE Root Mean Square Error.

RNN Recurrent Neural Network.

SARIMA Seasonal Autoregressive Integrated Moving Average.

SBLT Sequence of Burst lengths and Times.

SES Simple Exponential Smoothing.

SMAPE Symmetric Mean Absolute Percentage Error.

SNI Server Name Indication.

SoC Security Operation Centre.

SOTA State of the Art.

SPI Shallow Packet Inspection.

SPLT Sequence of Packet Lengths and Times.

SST Scargle's Significance Test.

SVM Support Vector Machines.

TAP Test Access Point.

TCP Transmission Control Protocol.

TLS Transport Layer Security.

TOR The Onion Router.

TS Time Series.

TSA Time Series Analysis.

TTL Time To Live.

VoIP Voice over IP.

VPN Virtual Private Network.

Introduction

Network monitoring plays a crucial role in the overall computer security management [59, 90]. Compared to protections deployed on end devices (such as AntiVirus software), network-based systems can protect infrastructure against users' sloppiness, policy violations, or intentional attacks from the inside [59]. Central to this strategy are Intrusion Detection System (IDS) and Intrusion Prevention System (IPS). An IDS acts as a passive monitoring tool that analyzes network traffic for signs of malicious activity or policy violations, alerting administrators when threats are detected. In contrast, an IPS is an active extension of an IDS that not only detects but also attempts to block or prevent detected activities in real-time. These systems typically rely on signature-based detection, which matches traffic against known attack patterns (e.g., Snort or Suricata).

To achieve high-fidelity detection, these systems often employ Deep Packet Inspection (DPI). Unlike standard packet filtering that only examines network headers (such as IP addresses and ports), DPI inspects the actual payload of the packet. By reassembling data streams, DPI can identify specific applications, extract metadata, and detect malware signatures hidden within the application layer. However, the efficacy of DPI is intrinsically linked to the visibility into the packet payload.

However, maintaining network security has become increasingly challenging in recent years due to mass traffic encryption and consequent reduced visibility [223]. Because DPI cannot inspect encrypted payloads without a decryption proxy, its utility is severely limited. Implementing such proxies is often unfeasible in large-scale infrastructures [18], such as Internet Service Provider (ISP) networks, due to performance bottlenecks and privacy concerns. Moreover, the encryption of Transport Layer Security (TLS) certificates in TLS 1.3 [192], the deployment of encrypted Domain Name System (DNS) [88], and the Encrypted Client Hello (ECH) proposal [193] have removed the few remaining unencrypted fields essential for effective threat detection.

Therefore, packet-based monitoring loses a significant portion of its benefits, as processing individual packets becomes computationally expensive with diminishing returns. Modern IDS must instead work in conjunction with the remaining observable metadata, which primarily consists of unencrypted protocol fields, volumetric values, and packet

timestamps [247]. The combination of these features can be used to create a time series.

1.1 Motivation

Network traffic is inherently temporal, meaning that every packet or flow record is timestamped and carries numerical attributes (e.g., packet sizes, inter-arrival times). Both timestamp and numerical attributes reflect the underlying generating process. For this reason, representing traffic as a time series is a natural choice for modeling network traffic. Importantly, time already appears in many detection pipelines as an explicit feature [5, 145], which indicates that temporal information is useful in practice.

This matters in threat detection because many security incidents are not single events but temporal processes. For example, communication between an infected host and a Command and Control (C&C) infrastructure often exhibits time-dependent structure such as periodic beaconing, bursts, or stage changes over time [82, 186, 236]. In other words, the “shape” of traffic over time can carry a discriminative behavior. Consequently, applying Time Series Analysis (TSA) to time series obtained by monitoring network traffic can reveal complementary information for monitoring systems.

The statistical view of time series supports multiple monitoring tasks, such as network traffic classification and anomaly detection. However, most current approaches emphasize classification via so-called Sequence of Packet Lengths and Times (SPLT), where a short sequence of packets is fed into a Deep Learning (DL) model [5, 145]. While effective for supervised recognition typically do not treat TSA itself as a first-class tool for extracting and reasoning about temporal structure. As a result, systematically leveraging TSA for threat detection remains an underexplored direction in the current State of the Art (SOTA).

From an operational perspective, deployability is an additional gap. Even when TSA methods are promising in controlled evaluations, their practical adoption in real networks is not sufficiently addressed in the SOTA. The larger and faster the target network, the more constraints arise (e.g., throughput, memory, latency, and robustness to traffic drift), making deployment in high-speed ISP environments a crucial challenge. Therefore, designing TSA methods that remain effective under realistic operational constraints is essential for advancing network traffic threat detection.

1.2 Problem Statement

This dissertation thesis aims to identify suitable algorithms and mathematical methods for detecting security threats in high-speed encrypted network traffic using TSA. Therefore, the special focus is taken on the deployability of the proposed approaches in real-world networks. The topic uses the fact that the regularity and temporal characteristics of communication are significantly dependent on the types of applications and their use. Network communication often exhibits specific behaviors observable at the network layer.

The detection of security threats is of high importance for the defense of the network infrastructure. The TSA of time series from network traffic is a non-trivial task for which there is currently no reliable universal solution. Moreover, in a network traffic environment, there are typically large volumes of data that must be processed within a short timeframe (due to early detection), which complicates the solution.

The foundation will be research in the field of the possible use of TSA methods, probabilistic models, and artificial intelligence algorithms. While the goal will be novel detection methods deployable in the real (ISP) environment.

1.3 Goals of the Dissertation Thesis

The overall goal of the dissertation thesis is to use TSA of network traffic for the deployable detection of security threats. The overall goal can be divided into the following sub-goals:

1. Creation and description of time series from network traffic.
2. Research, design, and development of the detection of network security threats using TSA.
3. Evaluation of the deployability of proposed detection methods.
4. Creation and publication of network traffic time series datasets for the research community.

1.4 Overview of the Dissertation Thesis

This dissertation proposes deployable TSA approaches for detecting threats in high-speed encrypted ISP traffic by shifting from packet payload inspection toward temporal patterns of network traffic. The overview of the dissertation thesis is shown in Figure 1.1.

In the dissertation thesis, three types of time series were proposed: *Single flow time series*, *Flow time series*, and *Aggregated time series*. These time series types have different properties, advantages, and disadvantages, which were described in the thesis author's paper [A.1]. Moreover, each time series type enables the creation of different methods resulting in threat detection using TSA.

The *Single flow time series* were analysed for threat detection as shown in the left pillar of Figure 1.1. First, in the thesis author's paper [A.2], a method was proposed for classifying network traffic and detecting security threats using the TSA of *Single flow time series*. This approach was further improved to be deployable in the thesis author's paper [A.3], where the NetTiSA flow was proposed. The universal detection of security threats using NetTiSA flow is possible even in high-speed ISP networks. Moreover, the NetTiSA flow achieves similar results to the best SOTA methods specially designed for the detection of one type of threat.

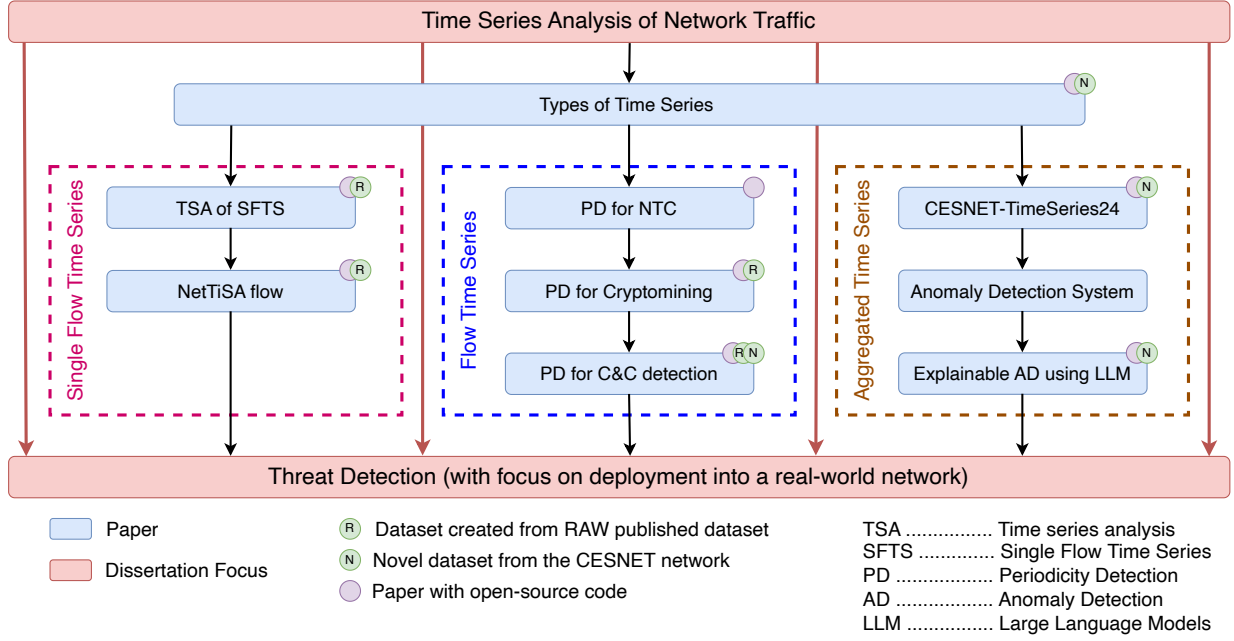


Figure 1.1: Overview of the dissertation

The *Flow time series* were analysed for threat detection as shown in the middle pillar of Figure 1.1. First, the periodicity detection method for network traffic classification of applications, services, and operating systems was proposed in the thesis author's paper [A.4]. This method was further improved in the thesis author's paper [A.5], with an application to threat detection, specifically in the context of cryptomining detection. And finally, the method was applied to the detection of botnets C&C communication, as presented in the thesis author's paper [A.6].

The *Aggregated time series* were analysed for threat detection as shown in the right pillar of Figure 1.1. First, the novel dataset, called CESNET-TimeSeries24, of aggregated time series of traffic from high-speed IPS network was created and published in the thesis author's paper [A.7]. The created *Aggregated time series* served for the evaluation of the detection of abnormalities using forecasting-based methods. In the thesis author's paper [A.8], an anomaly detection system was proposed, which successfully enables deployment of abnormality detection into high-speed networks. However, anomaly detection typically flags a large number of observations that are abnormal but not necessarily threats. Therefore, a novel approach published in the thesis author's paper [A.9] explains these abnormalities using Large Language Models (LLM).

1.5 Structure of the Dissertation Thesis

The thesis is organized into 7 chapters as follows:

Chapter 1 Introduction : Describes the motivation behind the thesis author's effort

and goals. There is also a list of contributions to this dissertation thesis.

Chapter 2 Background and State-of-the-Art : Introduces the reader to the necessary theoretical background and surveys the current state-of-the-art.

Chapter 3 Time Series from Network Traffic : Presents and describes the creation and properties of multiple network traffic time series, which are used in the thesis or are related to the thesis.

Chapter 4 Threat Detection using Single Flow Time Series Analysis : Describes a novel approach of threat detection based on TSA inside the IP flow exporter.

Chapter 5 Threat Detection using Multi-Flow Time Series Analysis : Gives insight into a novel approach for detecting security threats based on periodic behaviors within IP flow time series.

Chapter 6 Threat Detection using Aggregated Time Series Analysis : Describes a novel approach for detecting and explaining abnormalities within aggregated network traffic time series.

Chapter 7 Conclusions : Summarizes the results of our research, suggests possible topics for further research, and concludes the thesis.

1.6 Author's Publications Used in the Dissertation Thesis and His Contribution

The text in the dissertation thesis is based on the following published journal or conference papers. This section summarizes the thesis author's contribution to each paper.

Publication [A.1]: *Josef Koumar, Tomáš Čejka; Unevenly spaced time series from network traffic*

The author of this thesis created an experimental dataset and performed all the experimental evaluations described in the publication; besides, he also contributed to the writing. Tomáš Čejka helped with the writing of the initial draft and handled funding.

Publication [A.2]: *Josef Koumar, Karel Hynek, Tomáš Čejka; Network traffic classification based on single flow time series analysis*

The author of this thesis proposed the novel method presented in the paper, performed all the experimental evaluations described in the publication, and

also contributed to the writing. Karel Hynek (fellow PhD student at the time of writing this paper) helped design the evaluation methodology and contributed to the writing. Tomáš Čejka handled funding and performed a review of the initial draft.

Publication [A.3]: *Josef Koumar, Karel Hynek, Jaroslav Pešek, Tomáš Čejka; Nettisa: Extended ip flow with time-series features for universal bandwidth-constrained high-speed network traffic classification*

The author of this thesis proposed the novel method presented in the paper, performed all Machine Learning evaluations described in the publication, contributed to the writing, and handled the review process. Karel Hynek (fellow PhD student at the time of writing this paper) helped design the evaluation methodology, contributed to the writing, and helped during the review process. Jaroslav Pešek (fellow PhD student at the time of writing this paper) evaluated the deployment of the proposed method to the real-world networks, contributed to the writing, and helped during the review process. Tomáš Čejka handled funding and performed a review of the initial draft.

Publication [A.4]: *Josef Koumar, Tomáš Čejka; Network traffic classification based on periodic behavior detection.*

The author of this thesis proposed the novel method presented in the paper, created an experimental dataset, and performed all the experimental evaluations described in the publication; besides, he also contributed to the writing. Tomáš Čejka helped with the writing of the initial draft and handled funding.

Publication [A.5]: *Josef Koumar, Richard Plný, Tomáš Čejka; Enhancing DeCrypto: Finding Cryptocurrency Miners Based on Periodic Behavior*

The author of this thesis proposed the novel method presented in the paper, performed the experimental evaluations of the method, and also contributed to the writing. Richard Plný (master student at the time of writing of this paper) proposes a combination of novel methods with the DeCrypto framework, evaluates the performance of the Enhanced DeCrypto framework, and contributed to the writing. Tomáš Čejka handled funding and performed a review of the initial draft.

Publication [A.6]: *Dominik Oškera, Josef Koumar, Alžběta Pokorná, Kamil Jeřábek, and Tomáš Čejka; Botnet Detection Through Periodic Patterns in Command-and-Control Network Traffic*

Dominik Oškera (the thesis author's bachelor's student) captured malware traffic, finalized the dataset, and handled the modeling. **The author of this thesis proposed the methodology of capturing the dataset and detection of botnets, captured benign traffic, and also contributed to the writing.** Alžběta Pokorná (fellow PhD student at the time of writing the paper) performed a review of the methodology, dataset, and initial draft. Kamil Jeřábek (fellow PhD researcher at the time of writing the paper) contributed to the writing and reviewed the methodology and dataset. Tomáš Čejka handled funding.

Publication [A.7]: *Josef Koumar, Karel Hynek, Tomáš Čejka, Pavel Šiška; CESNET-TimeSeries24: Time Series Dataset for Network Traffic Anomaly Detection and Forecasting*

The author of this thesis proposes the time series metrics, designs and implements the architecture for collecting time series metrics, preprocesses data to the final version of the dataset, writes the initial draft, and handles the review process. Karel Hynek (fellow PhD researcher at the time of writing this paper) handled the application to the ethical committee, reviewed methodology and metrics, contributed to the writing, and helped with the review process. Tomáš Čejka handled funding. Pavel Šiška built the monitoring infrastructure for dataset capture.

Publication [A.8]: *Josef Koumar, Jaroslav Pešek, Kamil Jeřábek, Tomáš Čejka; Towards building network outlier detection system for network traffic monitoring*

The author of this thesis proposes the anomaly detection system, implements the code of the system, deploys the system into the network, evaluates the measurement, and contributes to the writing. Jaroslav Pešek (fellow PhD student at the time of writing the paper) reviewed the code of the system, helped with the analysis of the measurements, analysis of similar existing systems, and contributed to the writing. Kamil Jeřábek (fellow PhD researcher at the time of writing the paper) reviewed the code of the system, proposed a methodology of evaluation, and contributed to the writing. Tomáš Čejka handled funding.

Publication [A.9]: *Kamil Jeřábek, Josef Koumar, Jiří Setinský, Jaroslav Pešek; Explainable Anomaly Detection in Network Traffic Using LLM*

Kamil Jeřábek (fellow PhD researcher at the time of writing the paper) proposed the idea of the paper and methodology, helped with evaluation, and contributed to the writing. **The author of this thesis contributed to the ideation of the paper, reviewed the methodology, helped with the evaluation, and contributed to the writing.**

Jiří Setinský (fellow PhD student at the time of writing the paper) helped with evaluation and contributed to the writing. Jaroslav Pešek (fellow PhD student at the time of writing the paper) performed a review of the initial draft.

All co-authors of the thesis author’s publications used in the dissertation thesis have approved these contribution statements.

1.7 Ethics Statement

The advantage of real-world traffic data comes with a fundamental concern of privacy. For the creation of novel datasets within the thesis, only automatic data processing with immediate data anonymization was used. With this, in this thesis, it is declared that only anonymized data were analyzed, manually processed, or subjected to any procedures that could allow authors of the datasets to track users or reveal their identities.

The Committee for Ethics in Research of the Scientific Council of the Czech Technical University in Prague approved the publication of the CESNET-TimeSeries24 dataset under reference number 0000-10/24/51902/EKČVUT. The approval also includes a waiver of explicit user consent for publishing the dataset. Moreover, all users of the CESNET3 network agreed with the terms and conditions that define a monitoring process for optimization and improvement of services (including related research) and allow sharing of the data with third parties after anonymization (<https://www.cesnet.cz/en/gdpr>).

Therefore, all datasets created in this study were generated in accordance with this ethics statement, and none of the users were deanonymized during the dataset creation process. Moreover, none of the users can be deanonymized from the published datasets.

Background and State-of-the-Art

This chapter provides a theoretical background for network traffic monitoring, time series analysis, and an overview of related published works in the context of using time series analysis for network traffic monitoring and the detection of security threats. The most related papers and topics are grouped into several sections to cover the area systematically.

2.1 Time Series Analysis

In this section, the theoretical background for Time Series (TS) and Time Series Analysis (TSA) is provided to help the reader understand the terms used throughout the dissertation thesis. First, the definition of a TS is required, but primarily, Stochastic Process [256] has to be defined:

Definition 2.1.1 (Stochastic process). A stochastic process is a family of time-indexed random variables $Z(\omega, t)$, where ω belongs to a sample space and t belongs to an index set.

A time series is a realization of a specific stochastic process, such as the packet sizes of an Application Programming Interface (API) call to Facebook. A formal, distinct definition for TS is following:

Definition 2.1.2 (Time series). A time series is a sequence of observations taken through continuous measurements over time, such as the packet sizes of an API call. Formally:

$$\mathcal{T} = (x_{t_0}, x_{t_1}, \dots, x_{t_n}) \quad (2.1)$$

where $x_{t_i} \in \mathbb{R}^d$ is the i -th observation, $d \in \mathbb{N}_+$ defines the dimension of the time series, $t_i \in \mathbb{R}$ defines time of i -th observation, and $n \in \mathbb{N}$ defines number of observation in TS. A time series with $d = 1$ is called a univariate time series, while a time series with $d > 1$ is called a multivariate time series.

Therefore, a TS is a sequence of observations from some monitored process. Dimensions d in time series are typically referred to as time series variates or time series metrics. For example, the temperature in Celsius, wind in meters per second, and precipitation in millimeters are three time series metrics of a multivariate time series of weather monitoring.

The TS are also divided by the behavior of the sequence of times t . There are two types of time series defined below.

Evenly sampled time series

The evenly sampled time series are predominant in the literature; therefore, the term “time series” in the literature usually refers to evenly sampled time series [189]. Furthermore, evenly sampled time series can also be referred to as uniformly spaced or regularly sampled in the literature.

Evenly sampled time series naturally occur in many fields, such as finance [121, 208], retail [117, 222], economics [227], geology [183, 248], weather [63, 191], and medicine [103, 161]. There are many existing methods and tools for the TSA of evenly sampled time series because of their frequent occurrence.

The formal definition of an evenly sampled time series is:

Definition 2.1.3 (Evenly sampled time series). A series of values of a quantity obtained at successive times with equal intervals between them. Therefore, for time series $\mathcal{T}$ is true that:

$$\forall x_{t_i} \in \mathcal{T}, \forall t_i \in \mathbb{R}, \forall i \in \mathbb{N}, t_i = t_0 + i\Delta t \quad (2.2)$$

where $\Delta t = t_i - t_{i-1}$, x_{t_i} is the i -th observation of time series $\mathcal{T}$ and t_i is timestamp of the i -th observation. The substitution and division can be applied to time series timestamps as follows:

$$\frac{t_i - t_0}{\Delta t} = i \quad (2.3)$$

Therefore, the evenly sampled time series can be rewritten to $\mathcal{T} = (x_0, x_1, \dots, x_n)$.

The primary advantage of evenly sampled time series lies in the mathematical simplification of temporal dependencies, which allows for the direct application of efficient signal processing techniques and classical statistical models. Uniform spacing ensures that the sampling rate remains constant, facilitating the use of the Fast Fourier Transform (FFT) for spectral analysis without the need for complex interpolation or resampling [36, 228]. Furthermore, many advanced deep learning architectures, particularly Convolutional Neural Network (CNN) and standard Recurrent Neural Network (RNN), are inherently designed for grid-like data structures where the relative distance between observations is fixed [102, 139]. This regularity also simplifies the calculation of autocorrelation functions and the estimation of parameters in autoregressive models, as the lag operator remains consistent throughout the entire sequence [270]. Consequently, evenly spaced time series

minimizes the computational overhead and reduces the risk of bias introduced by non-uniform density in the temporal domain.

Unevenly spaced time series

The unevenly spaced time series are less frequent compared to the evenly spaced time series. Nevertheless, they naturally occur in many industrial and scientific domains, such as astronomy [51,204,250], natural disasters [137,179,214], medicine [221,230], economics [58], IoT data [15,87]. Furthermore, in literature, they can be referred to as an unequally spaced or irregularly sampled time series [31]. The formal definition of an unevenly spaced time series is as follows:

Definition 2.1.4 (Unevenly spaced time series). A series of values of a quantity obtained at successive times, where at least one interval between them differs from others. Therefore, it is true that:

$$\forall x_{t_i} \in \mathcal{T}, \forall t_i \in \mathbb{R}, \exists i \in \mathbb{N}, \Delta t \neq t_i - t_{i-1} \quad (2.4)$$

where x_{t_i} is the i -th observation of time series $\mathcal{T}$ and t_i is timestamp of the i -th observation. Therefore, the methods and tools for the unevenly spaced time series must work with both observations x_{t_i} and timestamps t_i simultaneously.

Unlike their evenly sampled counterparts, unevenly spaced time series present significant analytical challenges because the observational frequency often carries intrinsic information or reflects underlying stochastic processes. Conventional signal processing algorithms, such as the standard FFT, cannot be directly applied. Instead, specialized techniques like the Lomb-Scargle periodogram [141,203] are required to retrieve spectral information without introducing significant aliasing or bias [245]. In the context of modern machine learning, traditional architectures often fail to capture the temporal decay between observations, necessitating the development of time-aware models. These include Neural Ordinary Differential Equation (Neural ODE) or Continuous-Time RNN, which treat the hidden state as a continuous function of time to explicitly account for irregular gaps [120,194].

Time Series Analysis

TSA is a fundamental statistic utilized across diverse scientific domains to model the dynamics of sequential observations collected over discrete or continuous time intervals. By characterizing the temporal dependencies inherent in data, TSA provides essential insights into underlying patterns, enables the forecasting of future trends, and facilitates a deeper understanding of complex system behaviors in diverse fields, including economics, engineering, and environmental science.

The breadth of TSA encompasses various methodological pillars, including descriptive techniques, probability modeling, spectral analysis, and the study of bivariate and

multivariate processes. Advanced applications involve linear systems, state-space modeling—specifically the Kalman filter—non-linear dynamics, and structural volatility models [35, 41]. Within research contexts, such as the study of climate variability, TSA is particularly indispensable; it permits the extraction of nuanced signals and long-term changes that conventional, non-temporal statistical methods often fail to resolve [75].

The evolution of TSA has seen a transition from classical decomposition and exponential smoothing toward sophisticated modern frameworks. This includes the development of Autoregressive Integrated Moving Average (ARIMA), structural models, and stochastic volatility models, which are critical for predictive accuracy in finance and insurance [81]. Recent literature further underscores a shifting focus toward high-dimensional classification and prediction, reflecting the convergence of traditional statistics with machine learning to address contemporary analytical challenges [240].

Analytical Objectives and Categorization

The primary utility of time series analysis [189] lies in its ability to elucidate how variables evolve, revealing the dependencies between successive observations. This is critical because a robust understanding of temporal causes is necessary to distinguish genuine trends from stochastic noise.

Reliable TSA typically requires substantial data density; sparse datasets risk the identification of spurious periodicities or false trends induced by transient fluctuations. Methodological approaches are generally categorized based on the specific analytical objective:

- **Forecasting:** The most prevalent application, aimed at predicting future values based on the rigorous extrapolation of historical trends and seasonal patterns [13, 108, 112, 151, 269].
- **Anomaly Detection:** Often an extension of forecasting, this involves training models on historical norms to identify data points that deviate significantly from expected behavior [30, 113, 128, 198].
- **Classification and Clustering:** These techniques involve the categorization of time series into distinct groups based on shared characteristics. Classification can be distance-based or feature-based, often involving complex feature extraction procedures to compare the structure and composition of various series [2, 102, 231].

Descriptive and Explanatory Frameworks

Foundational to any investigation is descriptive analysis, which focuses on summarizing key features such as trends, cycles, and seasonal variations [19, 37, 154, 220, 234, 272]. While descriptive methods identify “what” is happening, explanatory analysis seeks to uncover the causal relationships and underlying drivers of the observed patterns. Often, exploratory analysis serves as a precursor to these steps, utilizing visualization and preliminary summarization to guide more detailed quantitative investigations. Usually, curve fitting is

employed to examine the mathematical relationships between variables within the data. By plotting data points along estimated curves, researchers can quantify interactions and project potential trajectories based on these established functional relationships.

Periodicity Detection and Spectral Analysis

The identification of recurring cycles is the central focus of periodicity detection [60, 84, 184, 190, 224, 258]. A cornerstone of this domain is the periodogram, as defined by Schuster [207], which estimates the spectral density of a signal to identify dominant frequencies. This concept has been refined through various methodologies, including Bartlett's procedure [25], the Welch method [24], the Laplace periodogram [138], and the Lomb-Scargle periodogram [245], the latter of which is particularly effective for unevenly sampled data.

Complementing spectral approaches is the Autocorrelation Function (ACF), which evaluates the correlation of a series with its own lagged versions. The mathematical foundations of these correlation coefficients were pioneered by Pearson [178]. Additionally, engineering-driven approaches, such as the Apriori algorithm, have been adapted for TSA. Originally used for frequent pattern mining via “join” and “prune” iterations [9], these techniques have evolved into dedicated frameworks for periodicity mining [174].

Data Transformation and Exponential Smoothing

Prior to analysis, data transformation is often necessary to isolate specific features or eliminate unwanted noise. Common techniques include log transformations to stabilize variance or linear transformations to facilitate easier interpretation.

Exponential smoothing represents one of the most enduring and effective methodologies for both interpolation and extrapolation. These methods vary in complexity:

- **Simple Exponential Smoothing (SES):** Suitable for data lacking clear trends or seasonality [173].
- **Double Exponential Smoothing (DES):** Often referred to as Holt's linear method, this extends SES to accommodate series exhibiting a trend [34].
- **Advanced Smoothing:** Further iterations include damped trend models and Triple Exponential Smoothing (Holt-Winters), which incorporate seasonal components to provide a comprehensive forecasting toolset.

2.2 Network Traffic Monitoring

Network traffic monitoring is a crucial task for network management and overall computer security [59]. In network management, it supports a wide range of key functions including performance monitoring, network troubleshooting, enforcement of Quality of Service and Quality of Experience policies, and strategic planning and optimization of network

resources [133]. Moreover, traffic monitoring systems help protect infrastructure by identifying user misconfiguration, policy violations, anomalies, and insider threats [64, 268]. Progress across all these research tasks relies heavily on the availability of high-quality, real-world data. The firewall can protect the infrastructure from external attacks, and antivirus software can detect threats on end-user devices. However, the firewall and antivirus software cannot detect all types of threats, such as complex attack chains with multiple attack vectors and malicious activities hidden within encrypted traffic [218]. Therefore, network traffic monitoring complements these protections and enhances their functionality [59].

Systems that use network traffic monitoring for the detection of security threats are called Intrusion Detection System (IDS) and Intrusion Prevention System (IPS), and they require additional infrastructure to build. First, the network traffic must usually be mirrored, for example, on the router using a mirroring port or a device called a network Test Access Point (TAP), or network traffic flows directly through some firewall device. In passive monitoring, traffic is mirrored to another device called a monitoring probe.

Packet-based Monitoring

Network communication is fundamentally discretized into packets, which serve as the basic unit of data transmission across a network. A packet consists of control information—contained in the header—and user data, known as the payload [1]. The header provides the necessary metadata for delivery, including source and destination IP addresses, protocol identifiers, and sequencing numbers, while the payload carries the actual application-level content.

Packet monitoring, often referred to as packet sniffing or capture, involves the real-time interception and logging of these data units as they traverse a network interface. This process is essential for performance troubleshooting, protocol analysis, and security auditing [59]. By capturing raw packets, administrators can reconstruct entire communication sessions to identify the root causes of latency or to detect unauthorized data exfiltration.

Deep Packet Inspection (DPI) serves as an advanced traffic analysis technique that operates at the Application Layer (Layer 7) of the International Organization for Standardization/Open Systems Interconnection (ISO/OSI) model. Unlike traditional Shallow Packet Inspection (SPI), which is limited to examining packet headers such as IP addresses and port numbers, DPI deconstructs the entire packet to analyze the payload in real-time [39]. This granular visibility enables the security infrastructure to distinguish between benign traffic and malicious activity that may masquerade as legitimate communication over standard ports.

Beyond its role in threat detection, DPI is essential for network management and policy enforcement within modern enterprise and industrial environments. It enables IDS and IPS systems to apply behavioral and signature-based matching across the data stream to mitigate data exfiltration and zero-day attacks [218]. Furthermore, network administrators leverage DPI for Quality of Service (QoS) management, prioritizing latency-sensitive ap-

plications, such as Voice over IP (VoIP), while throttling bandwidth-intensive, non-critical services.

Nevertheless, network traffic typically uses encryption for most of its content. Therefore, maintaining network security has become increasingly challenging in recent years due to the widespread use of mass traffic encryption, resulting in reduced visibility.

The encryption of Transport Layer Security (TLS) certificates by TLS1.3 [192], deployment of encrypted Domain Name System (DNS) [88], or the Encrypted Client Hello (ECH) proposal [?] removed the few remaining information essential for effective threat detection. Therefore, if possible, security managers are forced to deploy intermediate proxies to decrypt the traffic [175] and inspect it using DPI tools, such as Suricata¹. In such cases, deploying the intermediate proxy is much more intrusive than sending domain names and certificates in plaintext. The DNS, combined with the proxy, is an efficient solution for mid-sized, restricted networks. However, it does not scale to large provider-based networks, where threat detection is also desired [18].

Flow-based Monitoring

Each Internet Service Provider (ISP) needs to perform network monitoring and cybersecurity threat detection to enforce internal policies, protect infrastructure, and prevent line overload to maintain service [18]. The single IDS deployed at the ISP level can protect many users (even in the millions) and prevent or minimize the impact of attacks, such as Distributed Denial of Service (DDoS). In ISP deployment, the use of the intermediate proxy is unthinkable. It would outrage the consumers due to absolute payload availability; moreover, it is not feasible to process such a large amount of traffic transmitted over multiple 200 Gbps backbone lines with DPI. Therefore, large-scale infrastructures are often monitored using a flow-based approach, which represents the communication between two devices in the form of flows.

A basic IP flow is formally defined as a set of IP packets passing an observation point in the network during a certain time interval, sharing a set of common properties. According to the IETF standard [195], these properties are typically defined by the 5-tuple: source IP address, destination IP address, source port, destination port, and the IP protocol number. By aggregating packets into these flow records, the volume of data to be analyzed is significantly reduced compared to full packet capture, making it the de facto standard for high-speed backbone monitoring [59].

Flow-based network monitoring is crucial for maintaining security in large network infrastructures. Flow-based intrusion detection systems deploy a variety of classifiers and detectors for malicious communications. Naturally, the design of these detectors strongly depends on the information available in the flow. The flows can contain almost any data that can be extracted from the communication [90]. Modern flow export protocols, such as Internet Protocol Flow Information Export (IPFIX) or NetFlow Version 9, support templating mechanisms that allow users to define their data structures within flow records.

¹<https://suricata.io>

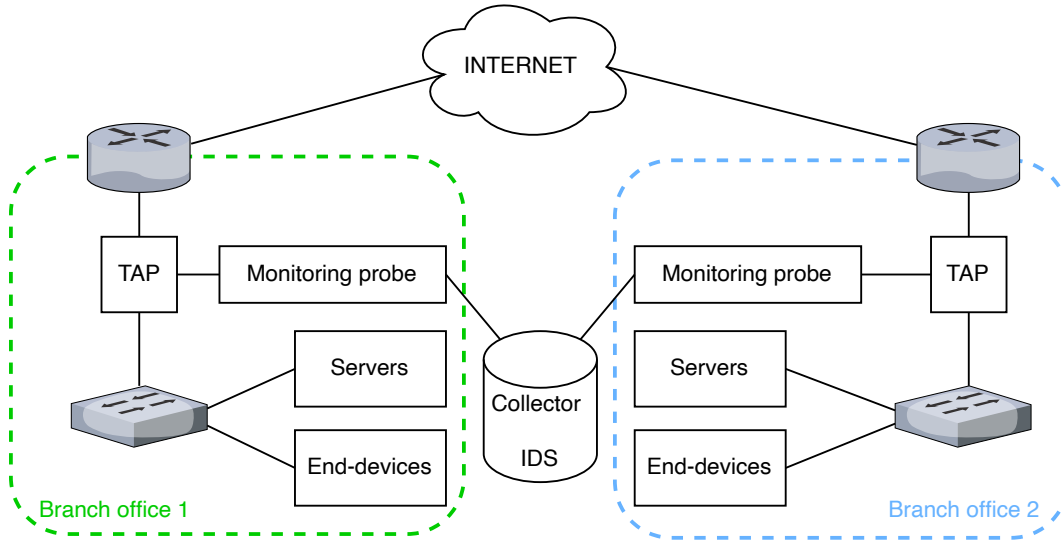


Figure 2.1: Example of the monitoring infrastructure with multiple monitoring probes

Thus, the flows are often extended to include various pieces of information helpful in intrusion detection.

Extended IP flows refer to flow records that go beyond the 5-tuple by incorporating additional metadata extracted from packet headers or limited payload inspection. For example, extensions may include Transmission Control Protocol (TCP) window size, TCP flags, and Internet Control Message Protocol (ICMP) types to identify scanning activities, or Application-Layer metadata such as Hypertext Transfer Protocol (HTTP) hostnames, DNS query-response pairs, and TLS handshake information (for example, Server Name Indication (SNI) or cipher suites) [247]. These extensions are vital for modern security analysis, as they provide the necessary context to detect sophisticated threats, such as botnet C&C communication or data exfiltration, without the computational overhead of full DPI [96].

In small networks, the monitoring probe can be part of the IDS system on the same server. However, this is usually not possible in large infrastructures because the network usually has a redundant connection to the internet. Thus, multiple monitoring probes are deployed in such a network, and the probes send IP flows to a server called a collector, which is part of the monitoring system. An example of such monitoring infrastructure is shown in Figure 2.1.

Network Traffic Monitoring for Threat Detection

Network traffic monitoring serves as a cornerstone for the proactive identification and mitigation of cyber threats. By systematically examining packets or IP flows, security systems can distinguish between legitimate operational traffic and patterns indicative of malicious intent [96]. Among contemporary methodologies, flow-based monitoring has

emerged as the preferred approach for high-speed networks due to its superior scalability and lower computational overhead compared to full packet inspection [59].

The analysis of IP flows allows security analysts to characterize the communication posture of a network beyond mere volume. By evaluating flow attributes such as duration, packet inter-arrival times, and byte distributions, analysts can detect a broad spectrum of threats. This includes volumetric DDoS, where multiple sources exhaust the resources of a target [187], and stealthy intrusion attempts, such as horizontal scanning or lateral movement, which often manifest as anomalous connection patterns to internal assets [218].

Effective detection relies heavily on establishing behavioral baselines. By constructing profiles of “normal” network activity, systems can trigger alerts for statistical deviations, such as an unprecedented data transfer from a high-value database to an unfamiliar external autonomous system. Such anomalies often indicate data exfiltration or the presence of a compromised internal host [247].

However, the efficacy of traditional flow analysis is increasingly challenged by the omnipresence of end-to-end encryption (e.g., TLS 1.3), which prevents inspection of the application payload. To overcome this visibility gap, modern detection frameworks have shifted their focus toward Machine Learning (ML) and Deep Learning (DL) architectures. These models leverage the statistical properties of encrypted traffic—such as packet size sequences and entropy—to identify malware communication and botnet Command and Control (C&C) channels without requiring decryption [20, 236].

2.3 Network Traffic Classification

The classification of network traffic to gain insight into encrypted traffic is a well-established domain [59, 96, 223, 247]. The purpose of the research includes, for example, detecting security threats in IDS systems [82, 125, 131, 186, 215, 236, 254], identifying Virtual Private Network (VPN) or The Onion Router (TOR) traffic [52, 57, 109, 165, 275], and classifying services in TLS or Quick UDP Internet Connections (QUIC) traffic [4, 5, 145, 147].

The flow-based IDS deploy a variety of classifiers and detectors of malicious communications. Naturally, the design of these detectors strongly depends on the information available in the flow. Since a flow record has a relatively loose definition [90], records can contain almost any data that can be extracted from the communication. Modern flow export protocols, such as IPFIX or NetFlow Version 9, support templating mechanisms that allow users to define their own flow extensions for network traffic classification. The flow extensions can be divided into three directions: 1. Extension for extracted unencrypted information, 2. Extension for packet sequences, and 3. Extension for precomputed features. These approaches are further described in the following sections.

Extension for extracted unencrypted information

Exporting unencrypted information from the packet payload is a crucial functionality of various flow exporters. It is common to export domain names transferred in DNS packets,

information from HTTP headers, and TLS handshakes, or simply just part of the payload. These payload fields enable a DPI detection approach even with the flow monitoring infrastructure [247]. However, each detection and classification task requires the extraction of different fields from the packets; thus, multiple flow extensions are required in practice, resulting in large telemetry records.

Due to its reliability, the unencrypted information extension is a popular data source in flow-based monitoring; however, it is unusable with encrypted traffic. In recent years, a trend has emerged in traffic encryption and the development of novel privacy-preserving protocols. For example, plain text DNS is being replaced by its encrypted versions, such as DNS over HTTPS (DoH) and DNS over TLS (DoT) [73, 88, 93]. Privacy-sensitive information in TLS handshakes is already sometimes encrypted using the ECH extension [242], which poses a challenge for traffic classification [211]. Since most traffic nowadays is encrypted, the importance of extracting unencrypted information is rapidly decreasing, and it is being replaced by other types of flow extensions that target encrypted traffic analysis.

Extension for raw packet information

The extension of flows for packet sequences embeds the raw packet-level information about ongoing connections into the flows. Typically, flows are extended for a Sequence of Packet Lengths and Times (SPLT) that aims to export the first n packets of flow as a sequence of packet lengths (or payload lengths), directions, and inter-packet times [6, 142, 145]. Each network classifier can then process the SPLT differently to maximize its accuracy. For example, network classifiers proposed by Vekshin et al. [246] or Hynek et al. [99] process SPLT by additional feature extraction (min, max, median of the sequence), which is then used by ML algorithms.

Other approaches feed SPLT directly into the ML algorithm. In their study, Luxemburk et al. [145] used SPLT of length 30 packets for fine-grained network classification with a large number of labels. They use information from SPLT (packet lengths, directions, and times) as three separate sequences and process them with a CNN with 1D convolutions. The proposed approach achieved excellent results in classification, as the convolutions extracted discriminatory features themselves. Moreover, they also demonstrated that tree-based ML algorithms can achieve similar accuracy even when raw SPLT is used as input.

Another use of SPLT was proposed by Chen et al. [44] and also by Shapira et al. [212]. Both approaches embed the SPLT sequence into a 2D image, which is then used by a CNN with 2D convolutions. The biggest difference between the two approaches is in the size of the input images used. Chen et al. [44] uses SPLT for only the first ten packets and achieves a high accuracy of 88.42% on the classification of network applications. Shapira et al. [212] deal with VPN detection tasks and use SPLT of very long size. Their proposed detector achieved a 99.7% of accuracy; however, it requires 1400×1400 images as the input, which means that the size of each flow with SPLT sequence is at least 7.84 megabytes. Such large telemetry records are hardly imaginable in production deployment.

The SPLT can be considered deployable only when the exporter limits the sequence to contain data about several first packets, usually in the order of tens of packets. Thus, the

flow exporter ipfixprobe limits the size of the SPLT sequence to 30 packets. Furthermore, the Cisco Joy² software can export SPLT sequences of up to 200 packets—with a default value of 50 packets. The short length then limits the SPLT’s applicability to only short connections. Therefore, researchers proposed extending the flows to include additional raw features, capturing information about packets that do not fit into the sequence. For example, Tropkova et al. [241] proposed to use a Sequence of Burst lengths and Times (SBLT), which carries the information about individual packet bursts (times and amount of transferred data). Nevertheless, even SBLT has its own length limit, and the aggregation of packets into the bursts loses some information about the exact timing of packets within the burst.

The extension for the traffic histogram, which was used by Hofstede et al. [89] in the HTTPS brute force detection, also contains information about packets from the entire connection, regardless of their length. However, similarly to SBLT, histograms do not convey information about the timing of individual packets; moreover, information about packet order is lost.

Naturally, the previously described flow extensions can be combined. For example, Luxemburk et al. [147] used SPLT and also traffic histograms in their QUIC classification task. Aceto et al. [6] and Wang et al. [253] combine SPLT even with unencrypted information from packet headers. The combination of payload bytes shows better performance. However, it also significantly increases network telemetry size by the first x bytes of payload per flow. For example, both approaches use the first 784 bytes of transport-layer payload. When combined with SPLT of length 32, the size of each flow is 1296 bytes.

Extension for precomputed features

The extension for raw-packet information allows flexibility in feature extraction, enabling each detector to compute its own specific discriminatory features. Nevertheless, flows with raw-packet information tend to be larger, as they carry unprocessed raw data. Therefore, some flow exporters perform feature extraction during the flow creation process and provide extended flows with already extracted features. An example of such an exporter is the CICFlowMeter³ that extends each flow with 80 statistical features—mainly mean, standard deviation, max, and min of multiple countable information from packets, such as the number of packets and TCP flags. These features are then used by multiple researchers in various network classification tasks [8, 56, 152, 213, 271].

Similarly, as CICFlowMeter, MontazeriShatoori et al. [158] created a DoHLYzer exporter⁴ that produces features directly within the flows, specifically for DoH detection task. These features, however, are entirely different from those of the CICFlowMeter. The DoHLYzer and CICFlowMeter exported features are thus not universal. The inflexibility of the extensions for precomputed features then forces detectors to use non-optimal features, drastically degrading their performance.

²<https://github.com/cisco/joy>

³<https://github.com/ahlashkari/CICFlowMeter>

⁴<https://github.com/ahlashkari/DoHLYzer>

The natural solution would be to export as many features as possible to cover a wide variety of detection/classification use-cases. However, a large feature vector does not maintain the benefit of reduced flow size. For example, the list of discriminative features published by Moore et al. [160] in 2005 contains 249 different features, which would result in extended flows with unacceptable sizes of almost 1000 bytes. Since 2005, various novel features have been employed to target new protocols and encrypted traffic analysis; thus, even a set of 249 features is far from complete.

There are possible approaches for decreasing the size of large feature vectors, such as 1. compressive traffic analysis proposed by Nasr et al. [167], 2. use of embeddings created by Neural Network (NN) instead of classical feature computation as proposed by Sungwoong et al. [267], 3. or feature dimension reduction techniques such as Principal Component Analysis (PCA) [265]. However, any flow exporter deploys these techniques to reduce the feature set size and, thus, the flow telemetry size. Moreover, all these approaches employ lossy compression, which cannot be fully reconstructed. The feature is then used in its compressed form, which is not understandable to people. Since people then analyze and inspect flows or alerts from the detectors, the lack of understandability then poses significant limitations in real-world deployment [243].

Time Series Analysis in Network Traffic Classification

The integration of TSA into network traffic classification is increasingly vital for identifying threats within high-volume, encrypted streams. Modern methodologies typically leverage ML and DL to process temporal sequences, moving beyond static feature vectors to capture the behavioral dynamics of network protocols.

The most prevalent representation is the SPLT, which transforms an unevenly spaced sequence of packet arrivals into a multivariate time series of packet lengths and inter-arrival times. By capturing the first n packets of a flow, SPLT provides a detailed “fingerprint” of application behavior that is both computationally efficient for flow exporters and highly discriminative for classifiers.

Sequence-based Deep Learning Approaches

Research into SPLT-based detection often focuses on leveraging the temporal dependencies of encrypted flows. Anderson et al. [16] utilized SPLT metadata to distinguish malicious flows without decryption, while Vu et al. [251] and Sadeghzadeh et al. [196] employed Long short-term memory (LSTM) networks to automatically extract time-dependent features from packet sequences. Similar multivariate time series approaches have been successfully applied to detect DoH tunneling activities by modeling the time differences between successive flow records [158].

Multimodal architectures have further expanded this domain. Aceto et al. [5] developed a framework for VPN classification that processes multimodal traffic data simultaneously. Similarly, Luxemburk et al. integrated 1D convolutional and linear layers to classify TLS and QUIC services by combining SPLT sequences with global flow statistics [143, 147].

Temporal Feature Extraction and Behavioral Analysis

TSA is also applied through windowed feature extraction and behavioral modeling. Bai et al. [23] proposed segmenting Internet of Things (IoT) traffic into fixed time intervals, extracting features such as packet counts and protocol distributions to feed a CNN-LSTM cascade model for device identification. Alternatively, Abdullah et al. [3] combined time series principles with fuzzy logic to detect anomalies, allowing the system to evolve its knowledge base as it observes new traffic patterns.

Periodicity Detection for Threat Identification

A specific subset of TSA in this domain focuses on periodicity detection to uncover automated malicious activity. Identifying recurring cycles in traffic has proven effective for detecting botnets, Peer-to-Peer (P2P) communications, and scanning activities [80]. For instance, Eslahi et al. [62] utilized decision trees to classify HTTP traffic based on observed periodic signatures to identify botnet C&C channels. Similar mining techniques for regional periodicity have been deployed to detect decentralized P2P botnets [186], while periodicity remains a core feature for distinguishing legitimate webmail from automated background traffic [205].

2.4 Network Traffic Forecasting

Network traffic forecasting is a critical area of research due to its importance in network optimization, resource allocation, and anomaly detection [66, 83, 172]. Various approaches have been developed over the years, leveraging both statistical and machine learning methods [66].

Currently, statistically-based methods are commonly applied in network traffic forecasting with many published approaches. Such statistically-based methods include Auto-Regressive Integrated Moving Average (ARIMA), Seasonal ARIMA (SARIMA), Exponential Smoothing (ETS), or Hidden Markov Models [43, 157, 162, 239]. However, there are several limitations of statistical-based forecasting [49, 66, 119]. Most of them rely on the assumption of linearity, which restricts their ability to model complex, non-linear relationships often present in real-world systems. Methods like ARIMA and regression models require stationarity in data, meaning trends and variance must remain constant over time, necessitating preprocessing that can complicate analysis. Moreover, these models are sensitive to parameter estimation, and their predictive accuracy can degrade in highly dynamic environments where relationships between variables fluctuate. Statistical approaches also struggle to handle multivariate interactions effectively, leading to oversimplified representations of intricate patterns. These limitations collectively underscore the challenges of applying statistical models in scenarios with complex, non-linear, or rapidly changing dynamics.

Therefore, deep learning has emerged in recent years as a transformative approach to network traffic forecasting. Primarily due to its ability to model non-linear relationships,

capture complex dependencies within data, handle non-stationary time series effectively, and provide superior scalability [66, 111].

Deep learning became increasingly valuable for forecasting, particularly with the introduction of Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), which mitigate the exploding and vanishing gradient problems that affected earlier Recurrent Neural Networks (RNN) [139]. These deep learning models have been successfully applied in the network traffic forecasting domain [54, 188]. Nevertheless, the rapid expansion of deep learning forecasting variants in recent years underscores the need for a comprehensive evaluation.

The first major enhancement of LSTM and GRU models was their integration with Fully Convolutional Networks (FCNs) [61, 114]. Moreover, Temporal Convolutional Networks (TCNs) and their combination with an RNN variant were successfully applied in network traffic forecasting [33]. These hybrid models gained popularity due to their ability to outperform classical approaches while requiring minimal data preprocessing and easily scaling to large time series datasets [115].

Moreover, more sophisticated deep learning models originally developed for image classification have also been applied in the time series forecasting area. Notable examples include Residual Networks (ResNet) [86]. Building on these architectures, additional models have been developed, such as InceptionTime [101], which integrates residual blocks with Global Average Pooling layers.

Finally, one of the most recent forecasting approaches uses Transformers, which have become a prominent tool in time series forecasting due to their capability to model long-range dependencies, critical for network traffic prediction [257]. However, traditional Transformers face challenges such as inefficiency and a lack of interpretability in capturing temporal dynamics. To address these limitations, several advancements have been proposed. For instance, ETSformer [259] introduces exponential smoothing and frequency attention mechanisms to replace conventional self-attention, enabling the model to capture both seasonal and trend components in a decomposable and interpretable manner. Similarly, models like Informer [276] and Autoformer [261] leverage sparsity and frequency-based approaches to enhance efficiency and adaptability in long-term sequence forecasting. Despite their success, recent studies question the necessity of complex Transformer-based architectures, with findings that simple linear models can outperform them in several scenarios [273]. Nevertheless, Transformers remain at the forefront of innovations in network traffic forecasting due to their ability to incorporate dynamic time-dependent features and extract meaningful patterns from large-scale, non-stationary data [123, 127].

The forecasting is usually done separately when multiple time series of multiple processes are present [98]. However, a recent study [159] also studied forecasting of groups of related time series, exploring how algorithms can leverage shared structure among series.

2.5 Anomaly Detection in Network Traffic

In the field of network security, the term most widely used is anomaly detection. However, the term is more general than the same term in statistics; it includes not only statistical anomalies but also undesirable traffic, such as that generated by malware, botnets, and other malicious activities. In this dissertation thesis, the focus is on statistical anomalies, but to avoid confusion, the term 'outlier' is often used. This field is theoretically well covered for its importance in many fields, i.e., [85, 116, 219].

Detecting such outliers in the network might be challenging; various approaches have been explored. Classical unsupervised methods, i.e., isolation forests, are widely applied [244, 249]. Statistical methods like the Chauvenet's criterion and moving window procedures are specifically designed to identify outliers in non-normal distributions (which are common [28] in network traffic) [22, 155]. Additionally, time series forecasting models are leveraged for outlier detection [199]. A similar method is the use of the Local Outlier Factor (LOF) for detecting outliers based on the local density variations [14]. Researchers deal with noisy or incomplete data [197, 201].

The literature reflects a broad spectrum of algorithmic approaches tailored to specific traffic characteristics:

- **Clustering and Distributional Modeling:** Kumari et al. [130] demonstrated the efficacy of k -means clustering for real-time intrusion detection. Addressing the inadequacy of Gaussian models for bursty traffic, Simmross-Wattenberg et al. [216] proposed α -stable distributions, which better characterize the heavy-tailed nature of network marginals.
- **ML and Natural Language Processing (NLP) Integration:** Sophisticated models treat raw URLs as natural language, utilizing vector embeddings and optimized Support Vector Machines (SVM) to achieve high-precision classification [149].
- **DL Architectures:** Recent advancements include D-PACK, which integrates CNN with Autoencoders. This unsupervised approach enables early anomaly profiling by inspecting only the initial bytes of a flow's first few packets [97].

Despite the proliferation of these techniques, a significant gap remains between theoretical performance and practical utility. Most existing models are validated primarily on static, offline datasets and often fail to address the computational constraints and evolving dynamics inherent in production-scale deployments.

Time Series from Network Traffic

This chapter provides an in-depth description of time series created from network traffic, including an analysis and comparison of these time series. The approach for creating these time series from network traffic and their analysis was published in the thesis author's papers [A.1, A.4, A.7].

Generating Process

The generating process of a time series represents the underlying stochastic mechanism that controls the evolution of observations over time. Accurate characterization of the generating process is essential, as it ensures that the statistical inferences and forecasts derived from the data are valid and representative of the true population.

Network traffic is generated by multiple devices, each of which runs multiple programs or processes. Therefore, when creating the time series from network traffic, it is necessary to identify the correct generating process to capture the valid time series. The traffic must be divided so that only the examined process is present in the final time series. It is a simple task when the activity of a device or network is analyzed, because the IP addresses can be used as identifiers for dividing traffic. Nevertheless, the task becomes problematic when one specific process or application running on a particular device is directly analyzed. An example is the communication from Netflix about streaming some series. If access to the device were possible, creating a time series would be easy. However, for example, on the ISP level, network traffic is monitored without access to the device, and the entire network's traffic is captured.

Individual network devices are identified through the use of IP addresses, while specific registered or well-known application protocols are distinguished via transport layer ports. These features enable the classification of analysed traffic into *Network dependencies* proposed in the thesis author's paper [A.4] as follows:

Definition 3.0.1 (Network dependency). The *Network dependency* is defined as long-term communications between device pairs where a service is provided by one node to another.

3. TIME SERIES FROM NETWORK TRAFFIC

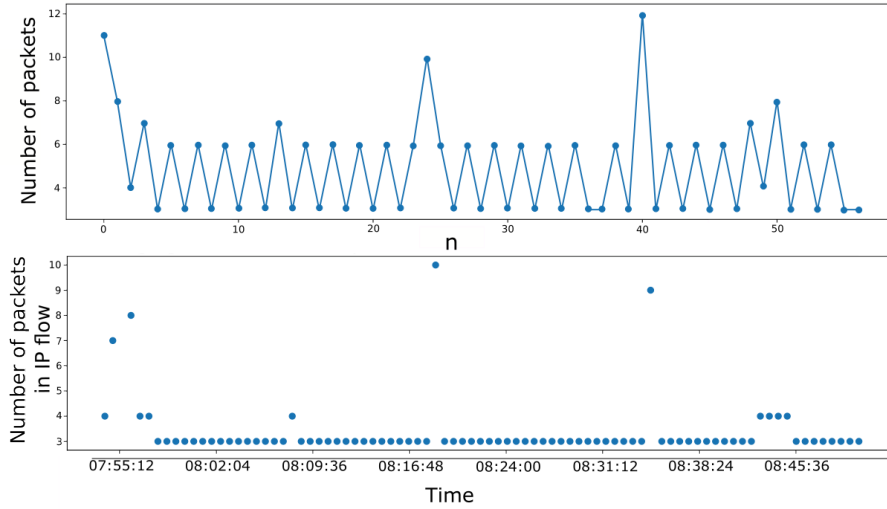


Figure 3.1: Comparison of evenly spaced time series (top graph) and unevenly spaced time series (bottom graph)

By leveraging these dependencies, the corresponding network traffic is merged into a unified time series. However, significant challenges arise during this process. In many architectures, multiple web servers may share a single public IP address, or conversely, several devices behind a Network Address Translation (NAT) gateway may communicate with the same destination server. Consequently, the segmentation of traffic into distinct network dependencies may fail, leading to the erroneous mixing of unrelated connections within the same time series. This phenomenon introduces substantial noise, as independent communications interfere with one another, thereby complicating subsequent Time Series Analysis (TSA). Furthermore, a single web application may comprise multiple concurrent processes that communicate under the same network dependency. Examples of such processes are independent webchat functions. These overlapping processes act as stochastic noise, further complicating the analytical task.

Once traffic is successfully partitioned, constructing a time series is a straightforward process. Since each packet or flow is associated with a transmission timestamp, its position within the sequence is precisely determined. This sequential mapping results in an *Unevenly spaced time series*. Alternatively, the network traffic (packets or flows) can be aggregated into fixed time intervals (windows), a common practice in the existing literature, for example, for network traffic forecasting. Aggregation of network traffic creates the *Evenly spaced time series*. For example, an evenly spaced time series generated with an aggregation interval of 60 seconds is shown in Figure 3.1. Explicit cyclic behaviors present in the unevenly spaced time series are often obscured or noised by the aggregation.

Therefore, the outcomes of TSA for evenly spaced time series are inherently dependent on the granularity of the interval. Moreover, the aggregation frequently results in the occurrence of zero values within the time series, which represent a significant limitation, as many standard mathematical methods and computational tools for time series are not

optimized to handle sparse data efficiently.

3.1 Types of Time Series from Network Traffic

Multiple types of time series can be created from network traffic. The types are designated by what represents one data point and by what data points have in common. In the thesis author's papers [A.1, A.7], four types were defined: *Packet time series*, *Single flow time series*, *Flow time series*, and *Aggregated time series*. Each type is defined and analysed below.

Packet Time Series

A *Packet time series* from network traffic is a time series where each data point represents a network packet. The *Packet time series* is a univariate time series with a variable number of bytes in the network packet. The number of bytes can represent either the size of the entire packet or the size of the payload within the packet. The time information $t_i \in \mathbb{R}$ of i -th data point is defined by the transmission time of i -th packet. So, *Packet time series* are inevitably unevenly spaced.

Definition 3.1.1 (Packet time series). The *Packet time series* is defined as a sequence:

$$\mathcal{T}_{\mathcal{P}} = (x_{t_0}, x_{t_1}, \dots, x_{t_n}) \quad (3.1)$$

where $t_i \in \mathbb{R}$ is time of observing i -th packet, $n \in \mathbb{N}$ is number of packets in time series $\mathcal{T}_{\mathcal{P}}$, and $x_{t_i} \in \langle 21, 65535 \rangle$ for the size of the entire packet or $x_{t_i} \in \langle 0, 65514 \rangle$ for the size of the payload. The size of one packet depends on the selected media layer. Mostly used Ethernet uses the Maximum Transmission Unit (MTU) [182] and is usually $x_{t_i} \in \langle 21, 1500 \rangle$ ($x_{t_i} \in \langle 0, 1479 \rangle$).

The advantage of *Packet time series* is that noise can be only entire data points because the value of a data point is always based on one packet. An example of noise can be retransmitted packets by the Transmission Control Protocol (TCP) protocol. There are indeed cases when the number of bytes per packet is aligned (padded) to some predetermined size, for example, in Virtual Private Network (VPN) traffic. However, in this case, the alignment is rather a process behavior than noise. As most of them, the examples of *Packet time series* shown in Figure 3.2 contain periodic behaviors. Both have data points, i.e., packets, that are likely to be noise.

Single Flow Time Series

Since a *Packet time series* may contain packets from any connection, it is useful to create a separate time series. Typically, in the *Packet time series*, multiple connections to the same service are present. Moreover, multiple services running on the same server are also usually present in one *Packet time series*. Therefore, the number of datapoints of the *Packet time*

3. TIME SERIES FROM NETWORK TRAFFIC

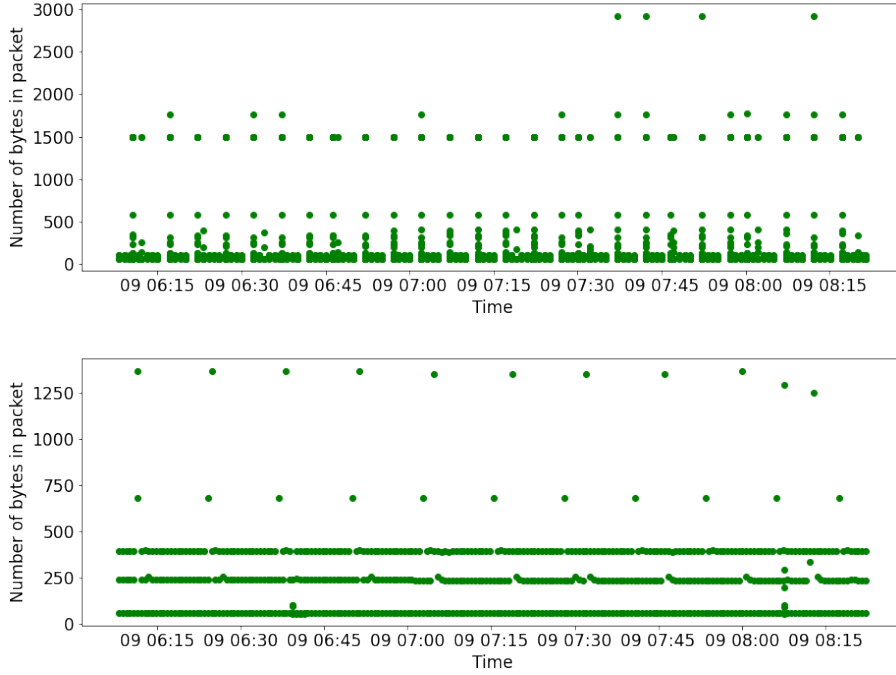


Figure 3.2: Examples of *Packet time series* with cyclic behaviors

series is large (thousands of datapoints in one minute), and packets from multiple services create a mess, resulting in a noisy time series. Both of these characteristics make analyzing packet time series challenging and consume a lot of time and resources. Therefore, the thesis author's paper [A.1] proposes a *Single flow time series* which addresses these challenges.

Definition 3.1.2 (Single flow time series). A *Single flow time series* is a *Packet time series* of one IP flow, i.e., a single connection that is divided into multiple packets if the time window of the connection is longer than an active timeout.

Because a *Single flow time series* is a special case of a *Packet time series* with only one flow, it usually contains network traffic from one process, where the only noise that can occur is packet retransmission. Therefore, the noise in the *Single flow time series* is minimal. Moreover, the number of data points in this time series is relatively small. Both of these characteristics allow real-time analysis.

Flow Time Series

Even if the noise caused by multiple services is removed from the *Packet time series*, the major disadvantage—a large number of datapoints—remains present. Therefore, a longer analysis of communication using *Packet time series* is unfeasible.

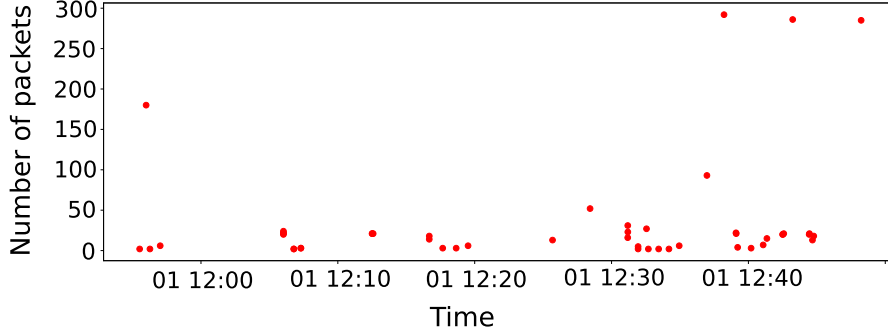


Figure 3.3: Example of *Flow time series* for network dependency 142.251.36.74(443)-10.0.0.10.

In the thesis author's papers [A.1, A.4], an approach was proposed that addresses these limitations of *Packet time series*. The approach defines the *Flow time series* with minimal noise based on splitting traffic into *Network dependencies*.

The thesis author's paper [A.4] proposes estimating the used service by using ports of the transport protocol in the TCP/IP model. The well-known and registered ports identify the services. Thus, an example of a network dependency is *192.168.1.1(53)-192.168.1.110* where IP *192.168.1.1* provides a Domain Name System (DNS) service to the IP *192.168.1.110*.

For unregistered ports, the network dependency is defined in multiple ways based on observed communication between two IP addresses. When two ports appear multiple times, network dependency is defined as a relationship between two IP addresses with those two ports, e.g., *78.128.191.25(5965-8884)-3.209.182.156*. If only one of the ports appears multiple times and the second one is changing with every flow record, then a network dependency is created based on the repeated port, e.g., *138.232.236.27(X-5228)-142.250.27.188*. Finally, if there is no stable port used for communication between two IP addresses, the network dependency is defined only by the IP addresses, e.g., *192.168.1.110()-192.168.1.111*.

Definition 3.1.3 (Flow time series). A *Flow time series* is defined as a time series for a network dependency by the following equation:

$$\mathcal{T}_{\mathcal{F}} = (x_{t_0}, x_{t_1}, \dots, x_{t_n}) \quad (3.2)$$

where $n \in \mathbb{N}$ is number of flows in time series $\mathcal{T}_{\mathcal{F}}$, $t_i \in \mathbb{R}$ is time of observing i -th flow (i.e., typically 0-th or n -th packet), $x_{t_i} \in \mathbb{R}^d$ is vector of values representing i -th flow, and $d \in \mathbb{N}_+$ represents number of time series variates.

As a *Packet time series*, the *Flow time series* is unevenly spaced. However, the *Flow time series* is a multivariate time series with multiple possible variates, including the number of packets in the flow, the number of bytes in the flow, and the duration of the flow. The example of a *Flow time series* is shown in Figure 3.3.

Aggregated Time Series

It is an inefficient use of unevenly spaced time series for some tasks. The presented time series types are limited in terms of the historical window that TSA of such time series can handle in real-time. Therefore, for tasks such as network traffic forecasting, where it is necessary to look into a longer history, the *Aggregated time series* is used.

Definition 3.1.4 (Aggregated time series). A *Aggregated time series* from network traffic is defined as a time series by the following equation:

$$\mathcal{T}_A = (x_{t_0}, x_{t_1}, \dots, x_{t_n}) \quad (3.3)$$

where $t_i \in \mathbb{R}$ is i -th time window, $n \in \mathbb{N}$ is number of datapoints of time series $\mathcal{T}_A$, $x_{t_i} \in \mathbb{R}^d$ is a vector of values representing an aggregated network traffic in i -th time window, and $d \in \mathbb{N}_+$ represents number of time series variates. For each time window t_i where $i < n$ is true:

$$w = t_{i+1} - t_i \quad (3.4)$$

Therefore, *Aggregated time series* $\mathcal{T}_A$ is evenly spaced with time window the w which serves as an aggregation window. So, it can be simplified to $\mathcal{T}_A = (x_0, x_1, \dots, x_n)$ and time length of the time series $\mathcal{T}_A$ is equal to $n * w$. Moreover, for $x_i \in \mathbb{R}^d$ is true:

$$x_i = \mathcal{A}(y_0, y_1, \dots, y_m) \quad (3.5)$$

where $\mathcal{A}$ is an aggregation function, $m \in \mathbb{N}$ is the number of observations on network, $y_j \in \mathbb{R}^e$ is an j -th observation (i.e., network packet or flow), and $e \in \mathbb{N}_+$ is number of attributes of observations (i.e., network packets or flows).

An aggregation function $\mathcal{A}$ aggregates network traffic into a vector of length d , which contains statistical characteristics of traffic. Examples of such characteristics include the number of flows, packets, bytes, unique destination IP addresses, and the average Time To Live (TTL). An example of an Aggregated time series is shown in Figure 3.4.

Network traffic collected on the monitoring probe must be divided into time series to capture, ideally, only the traffic of the desired generating processes. When Aggregated time series are used, usually the traffic of entire devices or networks is modeled. An example of such a use case with aggregated time series is network traffic forecasting.

3.2 Comparison of Time Series from Network Traffic

Multiple time series representations can be derived from raw network traffic, distinguished by the nature of their data points and the logical grouping of the underlying data. The structural relationship between *Packet time series*, *Single flow time series*, *Flow time series*, and *Aggregated time series* is illustrated in Figure 3.5.

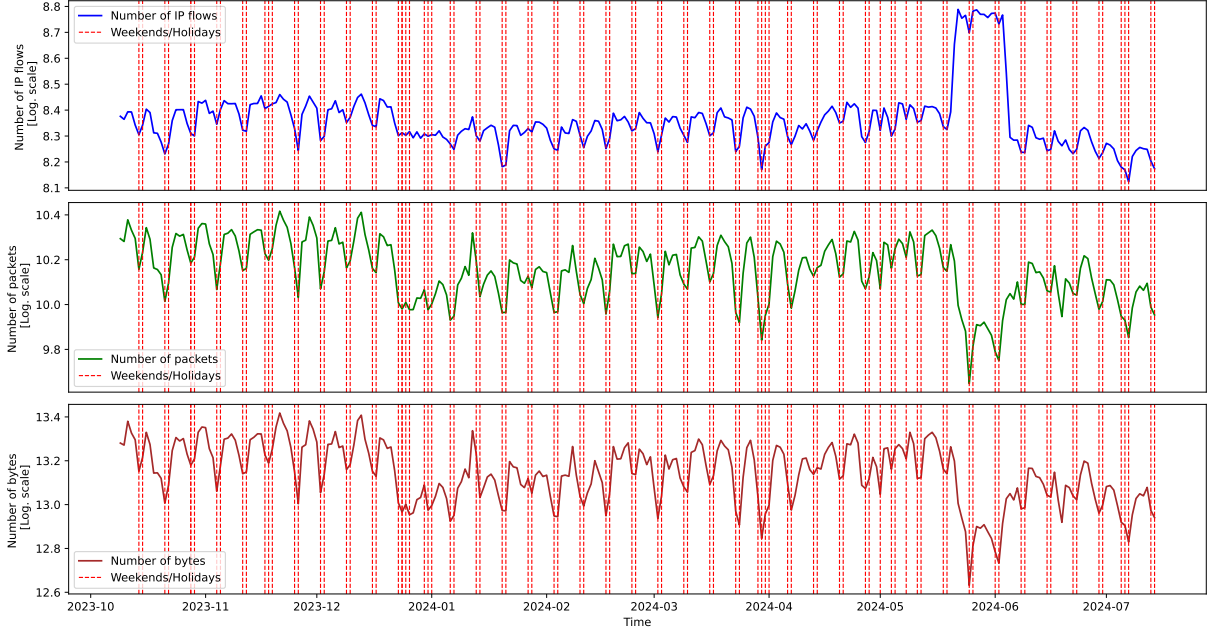


Figure 3.4: Example of *Aggregated time series* with three variates.

Inside an example network traffic, a clear periodic pattern of communication with some noise is present. In the *Packet time series*, this periodic pattern is clearly seen. However, the presence of the noise makes the analysis complex, and the number of data points makes the analysis time-consuming and resource-intensive.

From the *Packet time series*, four *Flow time series* were created based on different network dependencies. One of the *Flow time series* contains clear periodic behavior. However, in the real world, it usually has some noise in the values of the datapoints.

Each flow in the *Flow time series* can be interpreted as a *Single flow time series*. This time series type usually does not contain noise and is ideal for short-term analysis of network traffic.

The capacity of Aggregated time series to absorb stochastic noise while highlighting macro-level seasonal/periodic behavior is obvious. However, a significant amount of information can be lost using aggregation.

Unlike *Flow time series* and Aggregated time series, *Packet time series* is fundamentally resistant to datapoint noise, because every point represents a discrete, atomic event rather than a calculated summary. Furthermore, *Flow time series* is often ill-suited for analyzing specific traffic types such as VPN or NAT. In VPN communications, the forced discretization caused by the flow exporter's active timeout (typically every five minutes) creates artificial data points in the *Flow time series* that can be indistinguishable from large file transfers. Similarly, NAT traffic introduces significant noise into *Flow time series* data points by conflating disparate internal activities under a single external identifier.

Because *Packet time series*, *Flow time series*, and *Single flow time series* are unevenly spaced, temporal dynamics must be explicitly addressed during analysis. This is achievable

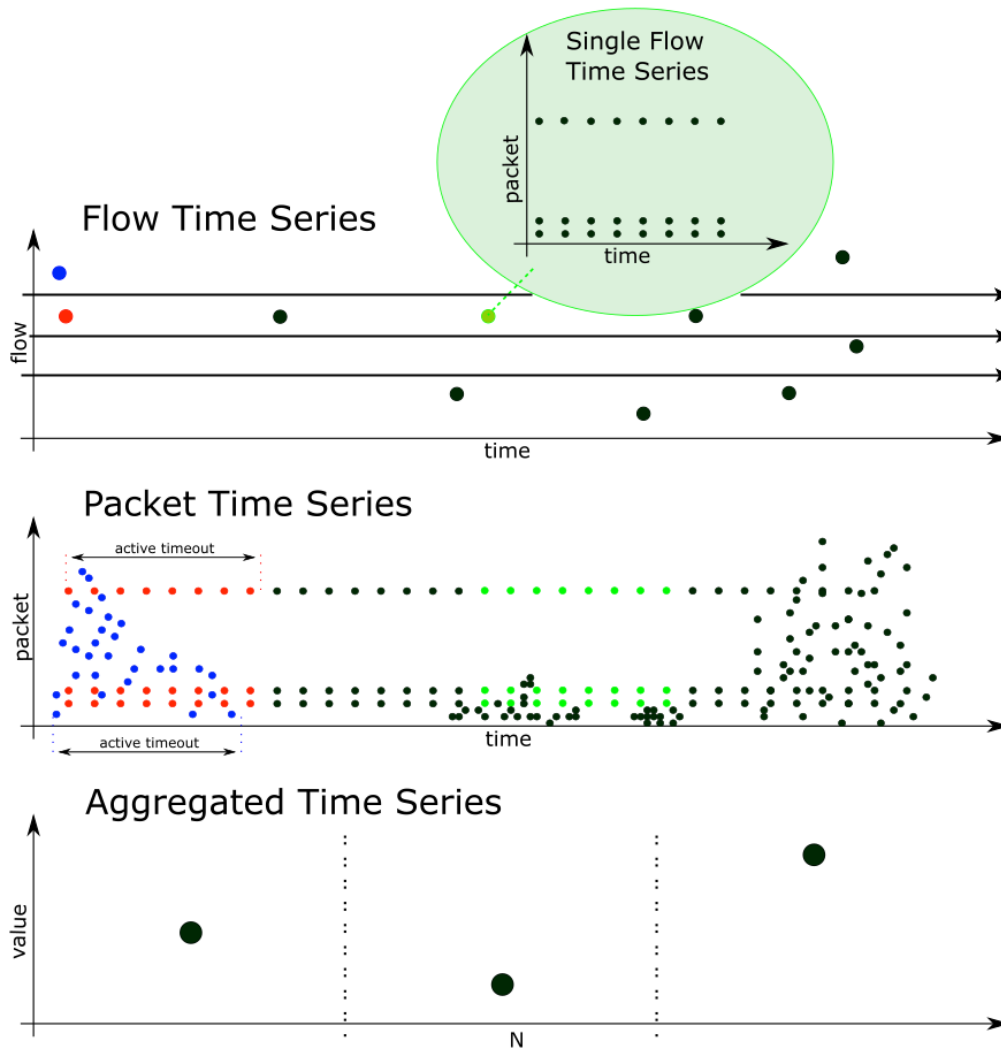


Figure 3.5: Relationship between types of time series from network traffic.

through two primary methodologies: first, by employing mathematical tools specifically designed for unevenly spaced time series that treat time as a fundamental coordinate [92,141,168,203,230]; or second, by transforming the data into a multivariate evenly spaced time series where time deltas are treated as an additional feature variable [158,196,251].

Threat Detection using Single Flow Time Series Analysis

In this chapter, the novel approach of obtaining discriminative features based on Time Series Analysis (TSA) of Single flow time series is evaluated. The novel feature vector was proposed in the thesis author's paper [A.2]. The feature vector achieves a similar performance to the best-performing related works on 15 datasets, making it a universal feature vector with high performance across multiple datasets and detection tasks. However, the limitations include the telemetry size of the flow with these features and the required resources for computing these features for each flow. These limitations were further addressed in the thesis author's paper [A.3] by proposing NetTiSA flow. The NetTiSA flow achieves similar performance across all tasks and datasets, with a significantly reduced telemetry size. Moreover, the required resources for computing NetTiSA flow are significantly lower, making possible deployment even in high-speed Internet Service Provider (ISP) networks.

4.1 Motivation

Network traffic monitoring provides information about activities in a computer network—an essential insight for maintaining service and security. As technology evolves, a classical approach using Deep Packet Inspection (DPI) is no longer feasible due to the increased need for privacy protection through encryption. Additional security features, such as the Encrypted Client Hello (ECH) [193], which encrypts even domain names, necessitate the development of new methods for monitoring and analysis to detect network threats and malicious activities.

Many research works [104, 134, 144, 153, 158, 181, 226, 254] thus utilize it in conjunction with machine learning for encrypted traffic classification to enhance visibility and identify encrypted malicious communication. Nevertheless, simple statistics such as the sum of transferred bytes and packets do not usually carry enough information for reliable traffic classification. The information about individual packet sizes, which has been found

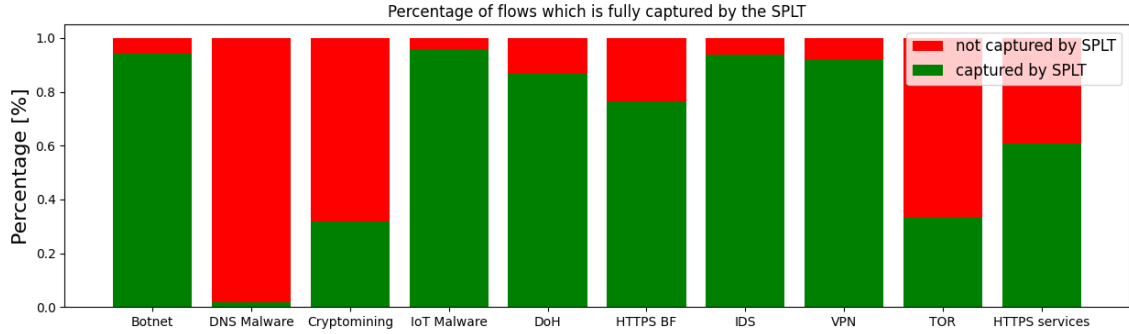


Figure 4.1: The shares of network flows in selected datasets that can (cannot) be captured by SPLT of length 30

extremely useful in previous research works [132, 145, 241], is lost when packets are aggregated into flow records. Therefore, several approaches have been proposed to extend flows in order to enhance classification performance. For example, flows are often extended by the Sequence of Packet Lengths and Times (SPLT) [145] or Sequence of Burst lengths and Times (SBLT) [241], and application-specific information fields [94, 95].

The SPLT and SBLT sequences significantly increase the amount of information that can be leveraged for classification. Still, they cannot carry information about all packets transmitted in the flows for practical reasons such as limited memory of the flow exporter or constraints on the flow record size. Therefore, SPLT often contains only the first n packets from the flow. For example, the Cisco joy exporter¹ exports detailed information (packet size, timestamp, direction) up to the first 200 packets in a flow, ipfixprobe flow exporter exports this information for only the first 30 packets.

Even these packet-extended flows thus still miss a lot of information when dealing with longer communications. The information loss is also demonstrated in Figure 4.1, which shows the percentage of flows that contain more than 30 packets in each selected detection task. It can be seen that the flow length depends on the classification tasks. In the case of The Onion Router (TOR), Cryptomining, and Domain Name System (DNS) malware, only a small portion of traffic is captured by SPLT. Nevertheless, all classification tasks would benefit from features that can capture flows longer than 30 packets. The benefit of a longer SPLT sequence can be seen in Figure 4.2, where each traffic type generates different patterns of packet sizes in time. These patterns can be captured by Time Series Analysis (TSA). Therefore, the thesis author’s papers [A.2, A.3] focus on proposing an additional feature set to extend IP flows with TSA to mitigate the information loss due to aggregation or limited SPLT or SBLT sequence size and propose a universal classification approach.

¹<https://github.com/cisco/joy>

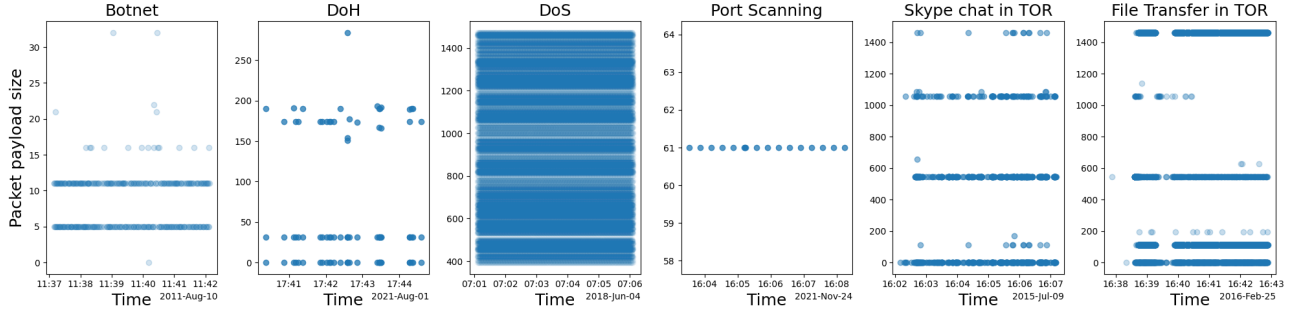


Figure 4.2: Examples of *Single flow time series* of different traffic types.

4.2 Exploration of features based on Time Series Analysis

The thesis author’s paper [A.2] proposes an approach to TSA in the flow exporter, creating a novel flow extension that can be used universally for network traffic classification. In this approach, time series are created from packets within a flow—the series payload sizes in bytes with the corresponding transmission timestamp to create a *Single flow time series*.

Time Series Analysis of Single flow time series

Single flow time series carries the behavior of the underlying process. These behaviors are discriminative from each other and can be used to identify the underlying process, as shown in Figure 4.2. In Figure 4.2, it is simple for the human eye to recognize different threats. The Machine Learning (ML) and Deep Learning (DL) can be used for recognition of these behaviors. For example, the SPLT represents the first n packets of these *Single flow time series* and was successfully used in related works for traffic recognition [5, 145]. However, only a small number of packets are usually exported in deployment, because the enlargement of telemetry is problematic in Internet Service Provider (ISP) environments.

The TSA of *Single flow time series* inside the flow exporter can be used for mining discriminative features about the underlying process without exporting a large number of packets in SPLT. The thesis author’s paper [A.2] evaluates the usability of such features for threat detection and traffic classification.

Features description

This section contains a detailed description of novel time series features. The features are organized into five categories: 1) statistical-based, 2) time-based, 3) frequency-based, 4) distribution-based, and 5) behavioral-based. Some of the proposed features for network classification have already been used in other fields of science, such as music classification [135, 206, 232]. A detailed description, including mathematical equations, of the entire feature set is published on the Zenodo platform [A.20].

Statistical-based features are based on the statistical evaluation of the sequence of observation x_{t_i} of the *Single flow time series*. The idea is a statistical description of data point deviation, i.e., statistical deviation of the packets' payload lengths. The statistical-based features contain 26 features, including *Mean*, *Median*, *Standard deviation*, *Pearson SK_1 skewness*, *Percent below mean*, *Galton skewness*, and *Average dispersion*.

Time-based features describe the time axis of the unevenly spaced *Single flow time series*. For computation time-based features, a two new sequences of times are used: a sequence of relative times $\hat{t}_i = t_i - t_0$, i.e., time from the beginning of a flow, and a sequence of time differences $\bar{t}_i = t_{i+1} - t_i$, i.e., time spaces between packets. The time-based features contain 9 features, including *Mean relative times*, *Median of time differences*, and *Duration*.

Distribution-based features are exported in the extended flow describes the distribution of data points in the *Single flow time series*. The distribution-based features contain 7 features, including *Hurst exponent*, *Normal distribution*, *Time distribution*, and *Stationarity*.

Frequency-based features stands on the idea to transform the *Single flow time series* into the frequency domain and analyse it. Based on recent research [79, 229, 260], the frequency domain has several advantages over the time domain. Frequency domain analysis is particularly useful for analyzing periodic behaviors because it enables the examination of individual frequency components and facilitates comparisons of the frequency content between different time series. Therefore, it is possible to obtain suitable features for classifying network traffic from the frequency domain. Since the *Single flow time series* are unevenly spaced, the Lomb-Scargle periodogram [245] must be used to transform the time series into a frequency domain. The frequency-based features contain 20 features, including *Power mean*, *Spectral bandwidth*, *Spectral flatness*, *Spectral periodicity*, and *Spectral zero crossing rate*.

Behavior-based features focus on describing the specific set of behaviors exhibited by the *Single flow time series*. The behavior-based features contain 7 features, including *Significant spaces*, *Switching ratio*, *Transients*, and *Periodicity*.

Feature Set Evaluation

The created feature vector contains 69 features that describe the behavior of the *Single flow time series* in depth. The thesis author's paper [A.2] evaluated the applicability of this feature set on most available datasets and tasks.

Dataset Selection

The publicly available datasets previously used or published in the network traffic classification domain were explored. Nevertheless, many datasets already contain precomputed

features and do not include raw packet-based data, which is necessary for the feature extraction based on TSA. Thus, the datasets of raw packet captures (PCAP files) were primarily considered. Together, the 15 well-known network datasets were selected, listed in Table 4.1, and processed using feature extraction. The processed datasets were published on Zenodo [A.20].

The selected datasets cover the most important traffic detection (binary) or classification (multiclass) tasks: 1) Botnet detection/classification, 2) Cryptomining detection, 3) DNS malware detection, 4) DNS over HTTPS detection, 5) DoS attack detection, 6) HTTPS Brute-force detection, 7) Intrusion detection/classification, 8) IoT malware classification, 9) TOR detection/classification and 10) VPN traffic detection/classification.

A baseline is necessary to evaluate the performance of the novel features. The baseline is the best-performing classifier for each concerned dataset. The recent classifier proposals (published after 2017) were searched using public research paper databases such as Google Scholar, IEEE Explore, and ACM Digital Library. More than 300 papers were explored and selected the best-performing proposals that met the following conditions ensuring fair comparability: 1) it was a flow-based method, 2) it uses the dataset as a whole and classifies all the dataset classes and types of samples, 3) does not use IP addresses as input features², 4) does not combine the concerned dataset with additional data. The selected best-performing proposals for both binary and multiclass versions of the classification tasks for each dataset are written in Table 4.1.

Evaluation Methodology

The features were evaluated by creating a novel classifier for each concerned network classification task. The classifier creation pipeline is the set of steps that creates the best final model. Initially, the published datasets were split into Train, Validation, and Test sets in a ratio of 60:20:20, while maintaining the labeling ratio as in the original datasets. Furthermore, some additional value sanitation is recommended. For instance, when handling a very short time series, it is necessary to handle “NaN” values. The “NaN” values were replaced by multiple values: for the distribution features with 0.5, for the frequency features with -1, and for the remaining features with 0. The source codes of the whole classification pipeline, including the pre-processing, are available at GitHub³.

In the validation phase, the optimal ML algorithm was selected. The 14 well-known ML algorithms were tested, including Random Forest, K-Nearest Neighbors (k-NN), or Support Vector Machines (SVM); nevertheless, the XGBoost algorithm achieved the best performance among all of the evaluated classification tasks. After selecting the algorithm, the best hyperparameters of the model were searched to achieve optimal performance on each dataset while avoiding overfitting. The `hyperopt` library [29] was used to tune the following hyperparameters: `n_estimators`, `max_depth`, `gamma`, `reg_alpha`, `min_child_weight`, and `colsample_bytree`.

²The concerned datasets are mainly lab-created; thus usage of IP addresses is not considered—in this case—as a good practice due to dataset overfitting as described by Behnke et al. [27]

³<https://github.com/koumajos/ClassificationBasedOnSFTS>

4. THREAT DETECTION USING SINGLE FLOW TIME SERIES ANALYSIS

Table 4.1: Description of *Single flow time series* datasets created from original datasets.

Binary classification						
Task	Original dataset	Labels [%]		Sizes of datasets		
		False	True	Train	Val	Test
Botnet	CTU-13 [74]	88.57	11.43	194,928	83,540	50,000
Crypto	CESNET-MINER22 [180]	64.97	35.03	1,417,433	607,470	1,075,576
DNS Mal.	CIC-Bell-DNS-2021 [152]	96.04	3.96	3,761	1,613	1,000
DoH	CIC-DoHBrw-2020 [158]	69.48	30.52	623,952	267,408	100,000
	DoH Real-world [106]	1.52	98.47	3,564,331	1,527,571	500,000
DoS	Bot-IoT [125]	97.40	2.59	2,106,874	902,946	1,000,000
BF	HTTPS Brute-force [146]	95.71	4.29	646,366	277,014	100,000
IoT Mal.	Edge-IIoTset [65]	92.95	7.05	827,557	354,668	250,000
	IoT-23 [71]	24.56	75.44	2,492,447	1,068,192	500,000
	TON_IoT [163]	0.31	99.69	2,077,190	890,224	500,000
IDS	CIC-IDS-2017 [213]	77.60	22.40	1,182,652	506,850	500,000
	UNSW-NB15 [164]	96.50	3.50	954,370	409,016	584,309
TOR	ISCX-Tor-2016 [132]	98.99	1.01	19,200	8,229	11,756
VPN	ISCX-VPN-2016 [57]	90.54	9.46	135,715	58,164	50,000
	VNAT [109]	96.61	3.39	23,380	10,020	10,000

Multiclass classification				
Task	Original dataset	Sizes of datasets		
		Train	Val	Test
Botnet	CTU-13 [74]	106,212	45,519	25,000
IDS	CIC-IDS-2017 [213]	1,392,651	596,851	200,000
	UNSW-NB15 [164]	954,370	409,016	584,309
IoT Mal.	Edge-IIoTset [65]	827,557	354,668	250,000
	TON_IoT [163]	2,252,190	965,224	250,000
TOR	ISCX-Tor-2016 [132]	70,557	30,239	20,000
VPN	ISCX-VPN-2016 [57]	12,655	5,423	5,000
	VNAT [109]	23,380	10,020	10,000

The hyperparameter search was performed using the training and validation datasets. The best values of the hyperparameters were selected based on the *F1-score* measure on the validation dataset. The final performance of the classifier on each dataset was obtained by evaluating the model trained on the training part on the test part. The test part was not used during any stage of the classifier design, ensuring the fairness of model evaluation

Table 4.2: Final results (in %) of binary classification. The green-colored cells represent results where the proposed approach is significantly (by 1% or more) better than the best related work. Red-colored cells represent results where the proposed approach is significantly worse. Other cells represent similar results.

Binary classification				
Task	Dataset	Method	Accuracy	F1-score
Botnet	CTU-13	Stergiopoulos et al. [226]	99.85	99.90
		Proposed approach	99.98	99.93
Crypto	CESNET-MINER22	Plný et al. [181]	93.72	90.59
		Proposed approach	95.29	93.11
DNS Mal.	CIC-Bell-DNS-2021	Kumaar et al. [129]	99.19	99.20
		Proposed approach	100.0	100.0
DoH	CIC-DoHBrw-2020	Zebin et al. [271]	99.98	99.91
		Proposed approach	99.90	99.84
	DoH Real-world	Jeřábek et al. [104]	97.5	98.7
		Proposed approach	97.79	98.80
DoS	BoT-IoT	Shagiq et al. [209]	99.99	99.99
		Proposed approach	100.0	100.0
BF	HTTPS Brute-force	Luxemburk et al. [144]	99.93	96.26
		Proposed approach	99.99	99.83
IDS	CIC-IDS-2017	Agrafiotis [8]	98.5	95.4
		Proposed approach	99.89	99.75
	UNSW-NB15	Ding et al. [56]	92.39	94.39
		Proposed approach	98.49	98.50
IoT Mal.	Edge-IIoTset	Khacha et al. [118]	99.99	99.99
		Proposed approach	99.99	99.97
	IoT-23	Sahu et al. [200]	~ 96	~ 96
		Proposed approach	99.86	99.91
	TON_IoT	Dai et al. [53]	99.29	99.03
		Proposed approach	99.96	99.98
TOR	ISCX-Tor-2016	Sarkar et al. [202]	99.89	99.88
		Proposed approach	99.84	96.33
VPN	ISCX-VPN-2016	Aceto et al. [6]	93.75	91.95
		Proposed approach	94.35	95.48
	VNAT	Jorgense et al. [110]	–	98.00
		Proposed approach	99.98	99.72

on data that was not seen before.

4. THREAT DETECTION USING SINGLE FLOW TIME SERIES ANALYSIS

Table 4.3: Final results (in %) of multiclass classification. The green-colored cells represent results where Proposed approach is significantly (by 1% or more) better than best-related work. Red-colored cells represent results where Proposed approach is significantly worse. Other cells represent similar results.

Multiclass classification					
Task	Dataset	Method	<i>Accuracy</i>	<i>Macro F1</i>	<i>Weighted F1</i>
Botnet	CTU-13	Marín et al. [153]	99.72	76.04	–
		Proposed approach	99.73	82.79	99.73
IDS	CIC-IDS-2017	Kunang et al. [131]	95.79	–	95.11
		Proposed approach	99.93	83.23	99.92
	UNSW-NB15	Ding et al. [56]	90.39	79.64	–
		Proposed approach	95.60	40.22	95.08
IoT Mal.	Edge-IIotSet	Khacha et al. [118]	98.69	–	–
		Proposed approach	99.97	89.75	99.97
	TON_IoT	Tareq et al. [235]	98.5	–	98.57
		Proposed approach	97.53	81.02	97.51
TOR	ISCX-Tor-2016	Yang et al. [266]	96.04	–	95.97
		Proposed approach	95.48	79.87	95.20
VPN	ISCX-VPN-2016	Dener et al. [55]	89.29	87.83	–
		Proposed approach	94.80	91.21	94.77
	VNAT	Jorgensen et al. [110]	96	–	–
		Proposed approach	98.60	98.88	98.60

Results of Evaluation

The results of binary and multiclass classification are presented in Table 4.2 and Table 4.3. On most of the binary classification problems, the novel feature set achieved similar or better performance than the best-performing previous work. Moreover, the novel approach outperformed eight related works significantly (by more than 1%). However, on the TOR detection problem, the approach achieved a worse F1-score than the best classifier.

The differences between the TOR classifier published by Sarkar et al. [202] were investigated, and it was found that they used a specially tailored feature vector that also includes transport ports. Although transport ports are often used in network traffic classification, the approach intentionally avoids them to maintain the universality of the features. The classifier often tends to overfit the transport port features, which, in some cases, is not a

desired behavior.

When evaluating the performance of the multiclass classification, the approach also outperformed most of the best-performing classifiers. Specifically, in five out of eight cases, the approach achieved a classification performance increase of more than 1%. However, in TON_IoT and UNSW-NB15 cases, a slight decrease was observed.

The best-performing classifier of TON_IoT published by Tareq et al. [235] is based on a 2D Convolutional Neural Network (CNN) with very long packet-length data (SPLT with all packets from the connection) organized in the image. The SPLT data provide the classifier with an opportunity for high-quality feature extraction, enabling accurate classification. Nevertheless, the long SPLT cannot be exported in real-world deployment scenarios due to the technical limitations of the flow exporters.

Besides, Ding et al. [56] achieved better results with Intrusion Detection System (IDS) classification using the UNSW-NB15 dataset. According to the analysis, the better results are attributed to a high-class imbalance in the dataset. Ding et al. thus proposed techniques for dealing with the imbalance ratio between classes. In this case, the approach did not incorporate any imbalanced learning techniques into the classifier design pipeline to maintain comparability with previous works, as most related works do not employ such techniques.

As shown in Table 4.2 and Table 4.3, the proposed feature vector proved to be universal and performed well on all relevant tasks. The slight reduction in accuracy in some cases was expected since universal features cannot compete with specially tailored ones; however, the performance reduction is considered, especially in the case of TOR or TON_IoT, as a good tradeoff for universality and the possibility of deploying all the network classifiers behind a single flow monitoring device.

Feature importance

The average feature importance is presented in Figure 4.3. On average, the best features are basic statistical features and, surprisingly, the *Transient* feature, which is important mainly in Botnet and IoT malware detection. Furthermore, the average feature importance was computed for each category of features (see Figure 4.4), and most categories have equal average feature importance. Only the frequency-based category has a minor significance in the average. However, one of the frequency-based features, *Min power*, is in the top ten best features.

Discussion

The proposed feature vector achieves comparative performance with the best-performing related works across 15 datasets in both binary and multiclass classification. Moreover, the best-performing related works were typically designed specifically for the evaluated tasks. Therefore, they have a significant advantage compared to the proposed feature vector. Nevertheless, the proposed feature vector based on TSA of *Single flow time series* achieved comparable results, making it universal across tasks and datasets.

4. THREAT DETECTION USING SINGLE FLOW TIME SERIES ANALYSIS

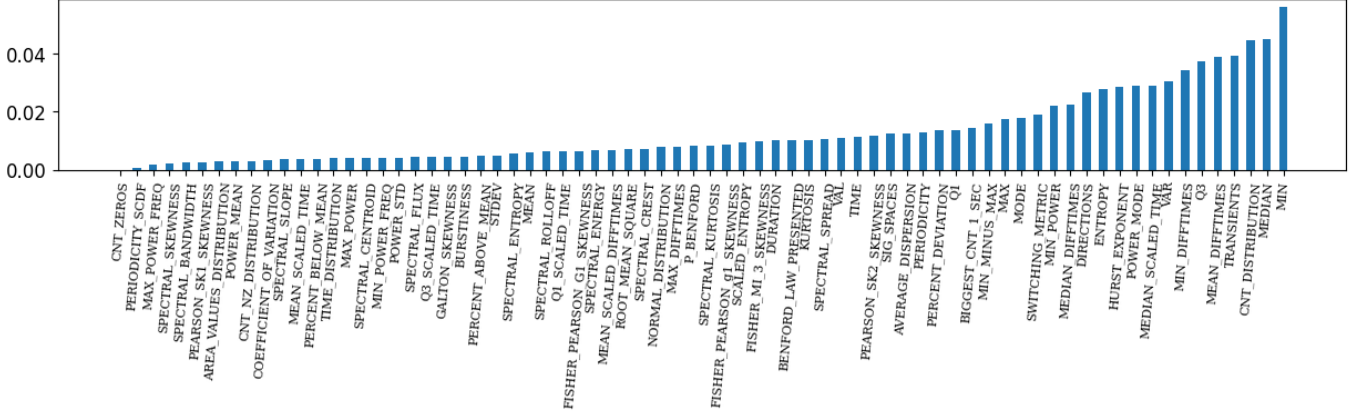


Figure 4.3: Average feature importance of the presented feature set

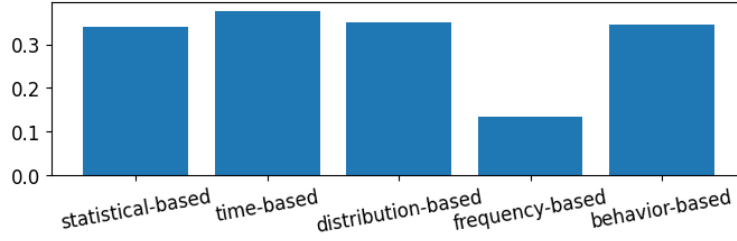


Figure 4.4: Average feature importance of each feature category

However, the proposed feature vector also has some limitations. The proposed feature vector contains 69 features, resulting in a flow record of 337 bytes. The classical flow has 61 bytes. Therefore, the proposed feature vector enlarges the flow size by more than 4.5 times. Network telemetry plays an important role in classifying network traffic. It is not enough to have a classification method that has excellent classification results. There is also the need to consider the network load of telemetry required for classification. Creating a balanced method in the size of network telemetry and classification results is necessary for practical reasons.

For this purpose, experimentation with feature set reduction was conducted to identify an ideally small number of features that perform well, as measured by the loss or gain in classification results. Furthermore, the feature set was reduced to a subset suitable for all presented problems, including multiclass scenarios.

Feature importance was evaluated on all 23 models presented in Table 4.1. It was observed that most binary problems primarily utilize statistical-based features. In contrast, frequency-based and distribution-based features were found to be the least important, yielding results that were approximately the same. However, for multiclass classification, frequency and distribution features were found to play an important role.

From the feature importance evaluation, 10 unique feature sets were obtained, each containing 10 features. These feature sets were used as input for the classification pipeline. Consequently, results were computed for each binary/multiclass problem and feature set;

specifically, $23 \times 10 = 230$ best models were created and evaluated⁴. Based on these results, the 10 feature sets were compared using statistical average and standard deviation. Similar results to those achieved with the original 69-feature set were obtained by the best feature set.

The selected best feature set contains the following features, sorted by importance: *Spectral kurtosis*, *Periodicity*, *First quartile*, *Benford’s law*, *Spectral energy*, *Median of time differences*, *Min*, *Third quartile*, *Min minus max*, and *Directions*.

This feature set was combined with fields from a classic IP flow that are not network or device-dependent, such as flow duration, the number of packets and bytes, and the number of packets and bytes in the reverse direction. The resulting classification performance reduction is shown in Table 4.4. The primary loss in classification results is attributed to the multiclass classification of Tor traffic, where the feature set performs significantly worse. Nevertheless, the average accuracy reduction across all use cases is less than 0.11%.

Table 4.4: Comparison of full feature set and best-performing reduced feature set. The comparison is done by average loss of classification Accuracy and F1-score (in %).

		Average	Accuracy	F1-score
Overall	Macro		-0.03 ($\pm$ 0.70)	-0.08 ($\pm$ 2.01)
Binary	—		-0.11 ($\pm$ 0.29)	-0.09 ($\pm$ 0.24)
Multiclass	Macro		0.02 ($\pm$ 1.32)	0.06 ($\pm$ 3.60)
	Weighed		0.02 ($\pm$ 1.32)	-0.19 ($\pm$ 0.99)

As shown, the feature vector can be significantly reduced to the 10 feature with a size of 101 bytes. The reduced feature vector does not lose significant performance across all datasets and tasks. Therefore, it is deployment-ready telemetry. It is true that a flow containing SPLT with 30 packets has 421 bytes, which is larger than a flow with a non-reduced feature vector. However, SPLT does not require computation and memory-intensive operations. The resource requirements are another limitation of this feature vector, which remains even for the best-performing reduced feature vector. This limitation makes it non-deployable in high-speed networks where a large number of flows occur per second, and was further addressed in the thesis author’s paper [A.3] by proposing NetTiSA flow.

4.3 NetTiSA flow

Features based on time series analysis are an ideal candidate for universal feature vectors as published in the thesis author’s study [A.2]. Nevertheless, the previous features were

⁴The total number of models evaluated in the validation phase for picking the best model was even higher, in the order of magnitude 2,500.

computationally and memory-intensive, which prevented their deployment in high-speed networks. Therefore, the thesis author’s paper [A.3] further explores the possibility of lightweight TSA that would enable deployment in all network infrastructures, including those with large and high-speed capabilities. The proposed feature set is exported in the novel extended flow, known as NetTiSA (Network Time Series Analysed) flow.

Features Description

The NetTiSA classification features are categorized into three groups based on computation. The first group of features is based on classical bidirectional flow information. The second group comprises statistical and time-based features calculated through the TSA of packet sequences. The third type of features can be computed from the previous groups (i.e., on the flow collector) and improve classification performance without impacting the telemetry bandwidth.

Flow features contains classical bidirectional flow information. However, the use of IP addresses and transport ports is explicitly omitted, since including them in the feature vectors is generally considered a bad practice, as described by Behnke et al. [27]. The flow features are: *Packets*, *Packets in reverse direction*, *Bytes*, and *Bytes in reverse direction*.

Single flow time series features are exported in the extended part of the flow. All of them can be computed (exactly or approximately) by online computation without saving the *Single flow time series* into memory, which is necessary for keeping memory requirements low. It contains the following features: *Mean*, *Min*, *Max*, *Standard deviation*, *Root mean square*, *Average dispersion*, *Kurtosis*, *Mean of relative times*, *Mean of time differences*, *Min from time differences*, *Max from time differences*, *Time distribution*, and *Switching ratio*.

Features computed at the collector are computed from the previous two groups before the classification itself. Therefore, they do not influence the network telemetry size, and their computation does not put additional load on resource-constrained flow monitoring probes. The *NetTiSA flow* combined with this feature set is called the *Enhanced NetTiSA flow* and contains the following features: *Max minus min*, *Percent deviation*, *Variance*, *Burtiness*, *Coefficient of variation*, *Directions*, and *Duration*.

Feature Extraction

Flows with all proposed features were extracted from every PCAP from datasets listed in Table 4.1. The flow extraction process was set with 300 seconds of active timeout and 65 seconds of inactive timeout⁵. Moreover, when a feature could not be computed (such

⁵If no packet is observed within the “inactive timeout” period, the flow is considered terminated. Flows longer than the “active timeout” are split and are exported every time this timeout elapses.

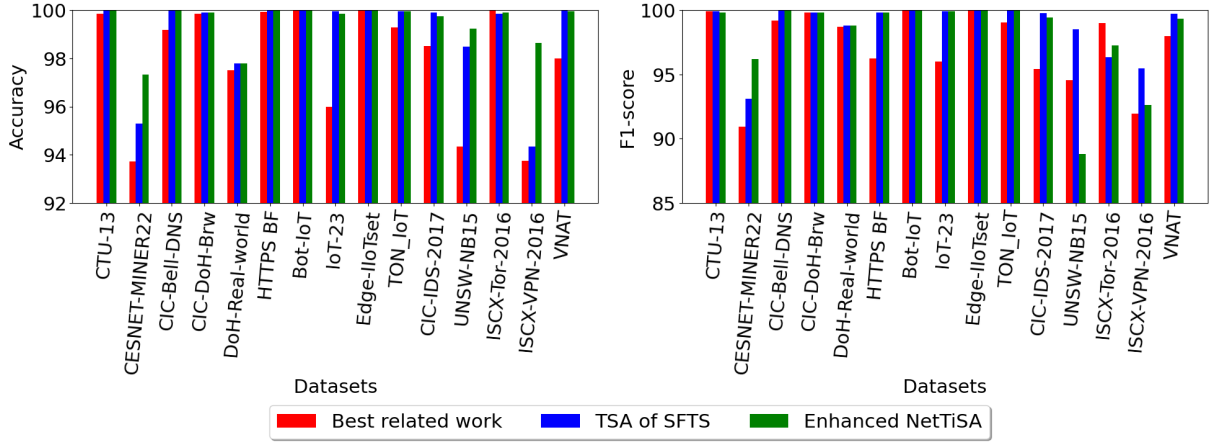


Figure 4.5: Comparison of binary classification by best-related works, complete feature set based on TSA of *Single flow time series* and Enhanced NetTiSA flow. The SFTS acronym in the legend stands for *Single flow time series*.

as Standard deviation or Switching ratio) due to a lack of data in short single-packet flows, its value was set to 0. Finally, the datasets of flow data extended for Enhanced NetTiSA features were split⁶ into a standard train, validation, and test parts, same as the previous approach (see Table 4.1), and the created datasets were published using the Zenodo platform [A.21] to facilitate reproducibility.

Results of Deployable Approach

The performance of the classifier trained on the novel Enhanced NetTiSA features is always compared with the best-performing classifier from the related work. Moreover, the performance is also compared with the original feature set.

Binary Classification

The results of binary classification are graphically presented in Figure 4.5. Detailed results can be seen in the published thesis author’s paper [A.3]. The classification based on the Enhanced NetTiSA feature set achieved mostly similar or better results than the best-performing classifier from related works. On six binary classification tasks, the Enhanced NetTiSA features performed significantly better, achieving more than a 1% increase in the accuracy measure or F1 score compared to related works. Nevertheless, with the TOR detection problem, Intrusion detection on the UNSW-NB-15 dataset yields a slightly worse F1 score than the best-performing classifier from related work.

Related works of intrusion detection on the UNSW-NB-15 dataset were investigated because the classifier published by Nawir et al. [169] performed significantly better in F1-score

⁶Stratified sampling was used to ensure that the original ratio of label classes remains in the split parts.

(by 5.75%). However, their accuracy is significantly worse than that of Enhanced NetTiSA (by 4.85%). The increased F1 score can be attributed to the application-layer-based features (such as those from HTTP headers) that would not be applicable in encrypted traffic analysis.

Also, the difference between the TOR classifier was investigated. The Dai et al. [53] extends flow data for the first 32 packets and also 600 bytes of payload. The payload information gives the classifier an advantage since it can fit the unencrypted Transport Layer Security (TLS) handshake data. Since the TOR usually does not use some common TLS extensions, e.g., Server Name Indication (SNI), the TOR TLS handshakes in the dataset are specific and easily recognizable by payload-based analysis. Nevertheless, the raw payload data gives Dai et al. an advantage, resulting in larger telemetry records.

The Enhanced NetTiSA discriminative performance is also compared with universal TSA of *Single flow time series* features proposed in the thesis author's paper [A.2]. In Figure 4.5 is visible that the results are mostly similar. The previous features show significantly better discriminative performance (in terms of the F1-score) in the ISCX-VPN-2016 and UNSW-NB-15 datasets. Nevertheless, the NetTiSA significantly outperformed the previous feature set on the CESNET-MINER22 dataset. The overall comparison of the binary classification is summarized in Table 4.5.

Multiclass Classification

The results of multiclass classification are presented in Figure 4.6. Similarly, as in binary classification, the Enhanced NetTiSA features proved to be discriminatory in the multiclass classification. It outperformed most of the best-performing classifiers proposed in related works. Specifically, in five out of eight cases, the Enhanced NetTiSA features achieved a classification performance increase of more than 1%. However, it observed a slight decrease in two cases—TON_IoT and TOR cases.

The best-performing classifier of TON_IoT published by Tareq et al. [235] is based on a 2D CNN with very long packet-length data (SPLT with all packets from the connection) organized in the image. The SPLT data provide the classifier with an opportunity for high-quality feature extraction, enabling accurate classification. Nevertheless, the long SPLT cannot be exported in real-world deployment scenarios due to the technical limitations of the flow exporters. Furthermore, although the Tareq et al. [235] achieves slightly better accuracy and weighted average F1-score (by almost 2.5%), they only achieved 52.2% macro average F1-score, which is 30.62% less than the classifier that uses the Enhanced NetTiSA feature set.

Similarly, as in the binary classification of TOR, Dai et al. [53] achieved a better Macro average F1-score with TOR of 86.77%, while the classifier trained with NetTiSA achieved only 79.87%. Nevertheless, the raw payload data results in larger telemetry records.

In comparing the Enhanced NetTiSA with the previous feature set, it can be seen that the novel Enhanced NetTiSA performs significantly better in four out of eight evaluated cases. In only one case—TON_IoT—the novel Enhanced NetTiSA features performed significantly worse (approx 1.5% lower Accuracy Weighted average F1-score). The overall

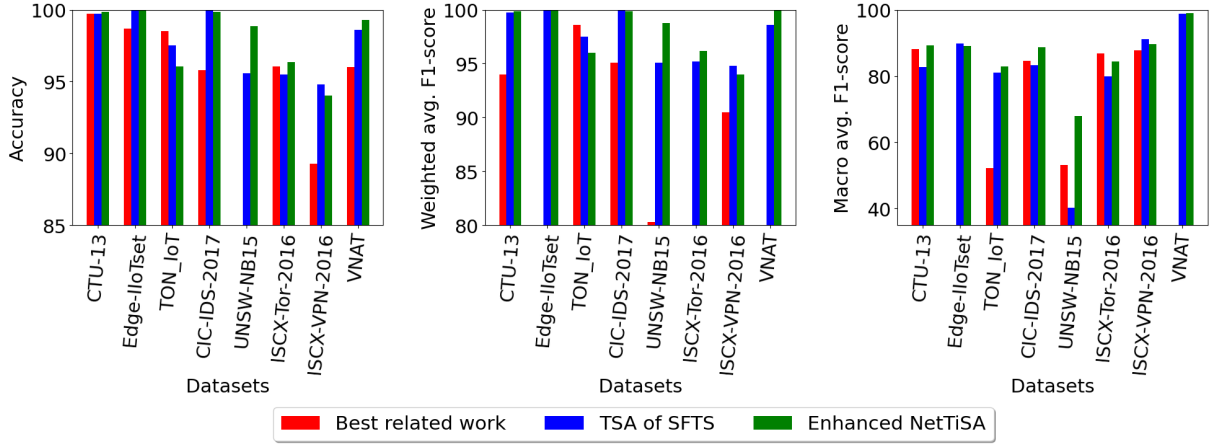


Figure 4.6: Comparison of multiclass classification by best-related works, complete feature set based on TSA of *Single flow time series* and Enhanced NetTiSA flow. The SFTS acronym in the legend stands for *Single flow time series*.

Table 4.5: Comparison of average results of binary and multiclass classification based on NetTiSA flows with classification based on all features generated using TSA of *Single flow time series*. Comparison is made by the average and standard deviation of each classification metric across all datasets (in %).

Binary classification	Accuracy	F1-score
NetTiSA	99.48 (± 0.840)	98.13 (± 3.167)
TSA of Single flow time series	99.02 (± 1.772)	98.74 (± 2.026)
Difference	0.45 (± 1.154)	-0.61 (± 2.681)

Multiclass classification	Accuracy	Weighted avg. F1-score	Macro avg. F1-score
NetTiSA	98.03 (± 2.098)	86.34 (± 8.267)	98.06 (± 2.186)
TSA of Single flow time series	97.70 (± 2.025)	80.87 (± 16.480)	97.59 (± 2.140)
Difference	0.32 (± 1.308)	5.47 (± 8.845)	0.46 (± 1.477)

comparison with the previous approach is shown in Table 4.5. It can be seen that, in multiclass classification, NetTiSA performs on average better in all performance metrics.

Discussion about Feature Discriminative Performance

Despite the expected performance drops, due to the universality of the feature set, the discriminative performance of Enhanced NetTiSA features is considered very good and

comparable with the best-performing proposals from the State of the Art (SOTA). In only four cases out of 23, the classifier performed more than 1% worse than the related work. Nevertheless, the reason behind the reduced accuracy was always investigated and explained. The higher accuracy in related works often stemmed from large telemetry sizes or higher complexity of feature computations.

Network Telemetry Size Comparison

The size of the input feature vector is crucial, especially in commercial deployments. A larger feature vector places additional stress on the monitoring infrastructure—the collectors and detectors must process more data. Moreover, in the case of flow monitoring infrastructure, the large amount of telemetry also utilizes bandwidth that could be otherwise used by customers [90]. Therefore, the sizes of network telemetry used as the input for the classification are compared across all the concerned tasks.

To form a reference point, the size of a classical IP flow record is established. A classical IP flow is defined as a combination of a flow key (Source IP, Destination IP, Source port, Destination port, Protocol) and a tuple of six values (the Number of packets and bytes in the flow in both directions, as well as the first and last packet timestamps). Thus, the transfer of this classical IP flow requires 21 bytes for the flow key and 40 bytes for the 6-tuple value, resulting in a total of 61 bytes.

Most approaches extend the flow to include additional data, which naturally results in a size increase. The flow records used by the best-performing classifiers on each evaluated classification task are analyzed to establish their input flow record size. When the datatypes were not specified, four-byte integers and four-byte floats are assumed for the record size calculation.

The results of the analysis are presented in Figure 4.7. It can be seen that NetTiSA requires almost the smallest flow records of 113 bytes (an increase of only 52 bytes compared to classical IP flow records). Most of the approaches require at least twice the flow extension. The only approach that utilizes smaller flow records is the one proposed by Jeřábek et al. [104], which compensates for the limited amount of information with active probing.

Discussion about Telemetry Size

The Figure 4.7 also shows that most approaches do not consider the telemetry size as an important factor. For example, the Jorgense et al. [109] on the VNAT dataset requires 3,673 bytes on a single flow record—their records would consume around 29 Gb of bandwidth per millions of flow. Such a bandwidth requirement makes the approach hardly deployable since the large ISP network can generate more than a million flow records per second. Compared to 29 Gb, NetTiSA flow consumes only 904 Mb per million flows. The traditional IP flow record requires only 488 Mb per million flows. Nevertheless, since the NetTiSA flow bandwidth fits into 1 Gbps lines, even with 1 million flows per second, its telemetry size is considered acceptable even in a large infrastructure.

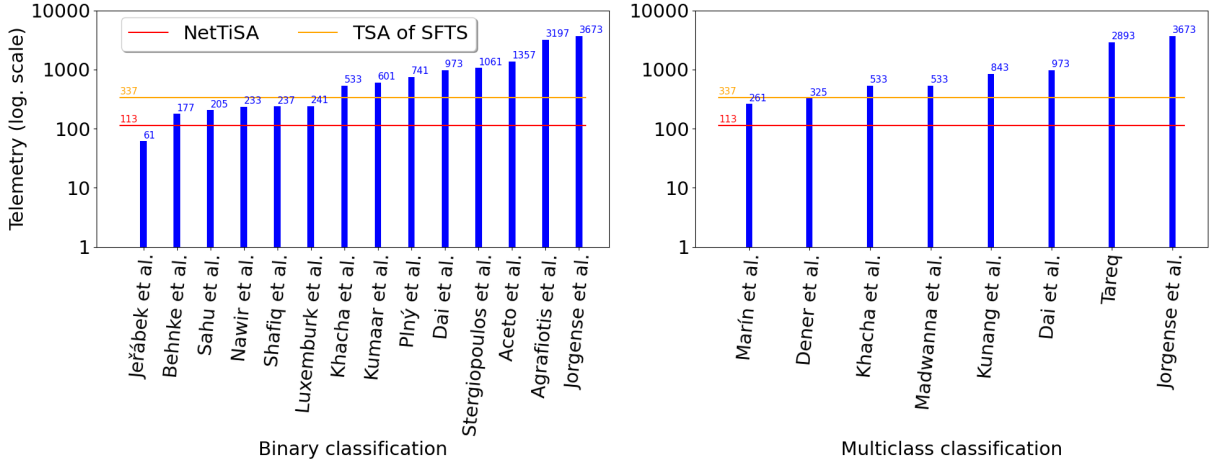


Figure 4.7: Comparison of telemetry size of best-related works, complete feature set based on Time Series Analysis of Single Flow Time Series and NetTiSA flow. The SFTS acronym in the legend stands for *Single flow time series*.

Impact of Features Calculation

To establish the feasibility of NetTiSA features in high-speed infrastructures, the performance impact of NetTiSA flow features calculation was also measured. Therefore, the feature calculation was implemented into the flow exporter ipfixprobe. The ipfixprobe is a high-performance flow exporter written in C++, that is capable of monitoring 100 Gbps backbone lines. It supports multiple input types, from traditional slow interfaces such as libpcap to high-speed Data Plane Development Kit (DPDK) [69] inputs for ISP-based monitoring.

The plugin-based architecture of the ipfixprobe was leveraged, and NetTiSA flow was implemented as the process plugin with the same name. For each flow, the online computation of each feature is performed by the plugin.

Regarding operation memory, 15 floats are stored in the memory for each flow by the NetTiSA feature computation, resulting in 60 bytes per flow. Moreover, with the arrival of each new packet, the plugin performs 23 mathematical operations and five comparisons. Additionally, 20 mathematical operations are performed before the flow is exported.

For the comparison, the ipfixprobe exporter was measured without the flow extensions (classical flows), and with three flow extensions: 1. the flow containing SPLT of length 30 packets (the *pstats* plugin in ipfixprobe), 2. proposed NetTiSA flow, and 3. an additional ipfixprobe process plugin was also implemented, that computes the previous time series analysis features. The plugins have been tested in two environments. In the lab-traffic environment, a stress test was conducted, and the limits of the exporters were evaluated using artificially generated data. In the real-traffic environment, anonymized PCAPs captured at the backbone ISP network were utilized, and the exporters were tested using real network data.

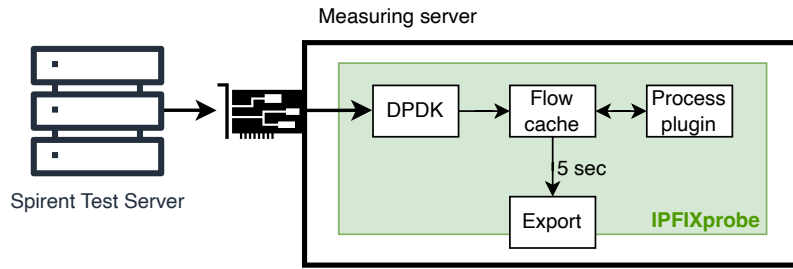


Figure 4.8: Measuring setup

Performance with Lab-traffic Environment

First, the performance is measured within the lab-traffic environment. For this purpose, a Spirent test server was utilized. Network traffic is generated by the Spirent test server according to specified parameters, thus performing stress tests of network devices. The packets were transmitted to the server with the ipfixprobe exporter, which was instantiated with the DPDK input plugin to allow high-speed packet processing. The measuring setup is shown in Figure 4.8.

As mentioned above, only the overall performance, which depends on the flow length, is of interest. Therefore, three constraints were defined to remove the influence of the input part of the exporter and flow cache as much as possible:

1. **Uniformly randomized packet sizes were used**, ranging from 80 to 1400 bytes, to maintain variable data for processing.
2. **The traffic was limited to only a single network flow.** The influence of other parts in the exporter (such as cache misses and collisions) is removed by the single flow stress test, allowing a focus on the processing plugin performance and memory requirements. The flow cache is forced to flush every five seconds to ensure that more than one flow is exported during the measurement. The measurement time is always 60 seconds; thus, the count of the exported flows should be exactly 12 ($60/5 = 12$). The number of packets/bytes in the flow is then regulated by setting frames generated per second.
3. **Only one input DPDK thread was set** since the measurement was performed in an environment where a specific single flow is always mapped to the same DPDK queue. Thus, the use of multiple threads would be redundant.

As can be seen in Figure 4.9, the exporting of NetTISA and SPLT features is comparable to classic IP Flow, and no performance drop is observed. On the other hand, the TSA of *Single flow time series* plugin lags behind (performance-wise) from every perspective. In the top left chart, the maximum throughput of the exporter was estimated based on the used plugin. The exporter is saturated relatively early in terms of frames per second—the input part of the exporter is saturated at approximately 500,000 frames per second.



Figure 4.9: Results of performance and memory requirements comparison using Spirent test server. The SFTS acronym in the legend stands for *Single flow time series*.

After that, no throughput gain can be seen. However, it can be observed that the TSA of *Single flow time series* causes saturation much earlier, and a throughput better than approximately 200 Mbps cannot be obtained.

In the top right chart, a metric called real-time per flow was used, which represents the time required for processing a single flow. This cannot be less than 5 seconds due to the flow cache timeout. As is shown, all plugins except TSA of *Single flow time series* behave reasonably—their processing time does not excessively exceed 5 seconds.

The bottom-left chart complements the second chart. It shows how many flows can be exported by the plugin—the maximum is 12, which means only the TSA of *Single flow time series* cannot export all available flows when frames per second are equal to or larger than 100,000.

In the bottom right chart of Figure 4.9, it is seen that the memory performance follows the computational performance regarding trends across all measured plugins. This measurement method was based on recording the maximum memory allocated by the exporter. Thus, besides the actual measured plugin, the overhead of the exporter was included. Since the processed data was identical, this overhead can be assumed to be comparable and thus detectable throughout the whole measurement.

Performance on Captured Real Traffic

For performance measurement on captured real traffic, three types of PCAP files were used for memory utilization: 1. Short flows – only flows with 30 or fewer packets, i.e., flows that are fully captured by SPLT, 2. Long flows – only flows with more than 30 packets, 3. Anonymized and payload-less CESNET2 captures – captures of a real office environment of a subnetwork inside the ISP CESNET2 network.

The performance comparison in processing time for each feature type computation is shown in Table 4.6. Expected behavior in performance on the real traffic captures can be observed. It must be noted that the processing time is influenced by the loading of packets from the PCAP file (the processing time of the same number of packets in deployment into a real network environment is much faster). The Classical IP flow has the best performance because the fewest operations are performed; the SPLT flow performs extra assignments into memory (max 30 times), and the NetTiSA flow performs several extra mathematical operations when a new packet arrives. Thus, these three types of flows are sorted by performance as: 1. Classical IP, 2. SPLT, and 3. NetTiSA flow. However, the difference in performance between them is not significant. Furthermore, the performance of the computation of features generated by the TSA of *Single flow time series* requires a significant amount of computation, resulting in a substantial decrease in performance compared to other types of flows.

Table 4.6: Performance comparison on real traffic by using captured PCAPs. The SFTS acronym in the legend stands for *Single flow time series*.

PCAP	Number of		Processing time [s] of			
	packets	flows	Classic IP	SPLT	NetTiSA	TSA of SFTS
Short flows	3.3m	24,6k	1.0	1.0	1.2	67.0
Long flows	139.2m	1.3k	40.4	44.3	45.0	877.1
CESNET2	26.6m	213.3k	7.7	8.1	9.6	523.1
captures	130m	1m	38.5	41.7	47.6	2563.9

The results of the memory requirements for each set of feature computations are shown in Table 4.7. Similarly to the performance measurement, packet captures (pcaps) from a real CESNET2 ISP network were used. The results are provided as the mean and standard deviation of memory usage per second. Several points can be observed from the results: 1. The number of packets in a flow does not influence the memory requirements of the feature computation for Classical IP, SPLT, and NetTiSA flows. 2. The number of flows does not highly influence the memory requirements of the feature computation for Classical IP and NetTiSA flows; however, the SPLT flow is influenced by it. 3. The TSA of *Single flow time series* feature computation has memory requirements that are highly influenced by the number of packets in the flow (saving the time series into memory requires dynamic

allocation). 4. The standard deviation shows that Classical IP, SPLT, and NetTiSA flows are highly stable in memory requirements, but the TSA of *Single flow time series* is highly unstable because of the freeing of large vectors of payload length and times after the exportation of the flow. 5. The difference between the memory requirements of the TSA of *Single flow time series* on Short and Long flows is substantial (ranging from 26.5 MiB per 1k flows to 5.24 GiB per 1k flows).

Table 4.7: The comparison of memory usage per second of input PCAPs processing. The comparison is done by mean and standard deviation. The SFTS acronym in the legend stands for *Single flow time series*.

PCAP	Number of		Mean memory usage per second [MiB] of			
	packets	flows	Classic IP	SPLT	NetTiSA	TSA of SFTS
Short flows	3.3m	24,6k	30.2 (± 0.0)	36.8 (± 3.1)	31.0 (± 0.4)	653 (± 615)
Long flows	139.2m	1.3k	30.1 (± 0.0)	30.5 (± 0.3)	30.2 (± 0.0)	6982 (± 7093)
CESNET2	26.6m	213.3k	30.1 (± 0.0)	40.9 (± 1.6)	31.5 (± 0.2)	2632 (± 4758)
captures	130m	1m	30.2 (± 0.0)	41.7 (± 0.8)	31.6 (± 0.1)	3847 (± 5855)

Discussion about the performance

Since the NetTiSA feature's computation performance was almost identical to that of the traditional flow computation, its performance is considered excellent. Moreover, the NetTiSA plugin has already been deployed in the production CESNET2 network, featuring eight monitoring probes that handle approximately 200,000 flows per second. Thus, the NetTiSA is deployment-ready for other flow-based monitoring infrastructure.

4.4 Key Findings and Contributions

This chapter presented the research on network threat detection using *Single flow time series* and the development of a high-performance, deployable approach known as NetTiSA. The key findings in this chapter are:

1. Standard flow-based monitoring often lacks sufficient information for reliable traffic classification because critical packet-level details are lost during aggregation.
2. Existing flow extensions like SPLT are limited by memory constraints and often fail to capture information for longer flows, which significantly impacts tasks like TOR or Cryptomining detection.

3. A universal feature set containing 69 TSA features was shown to outperform related works by more than 1% in eight binary classification tasks and six multiclass classification tasks.
4. Through experimentation, the feature set was successfully reduced to a subset of 10 ideally small features that maintain high performance with an average accuracy reduction of less than 0.11%.
5. The proposed NetTiSA flow extension is highly efficient for high-speed networks, requiring only 113 bytes per record and 60 bytes of operational memory per flow.
6. Performance tests in 100 Gbps environments demonstrate that NetTiSA feature calculation is stable and does not cause performance drops compared to classical IP flows.
7. The NetTiSA approach is deployment-ready and has been successfully integrated into the production CESNET2 network, handling approximately 200,000 flows per second.

The results in this chapter suggest a significant advancement in encrypted traffic monitoring. By leveraging TSA, network security tools can regain visibility lost due to encryption without the prohibitive computational costs of traditional TSA methods. The universality of the NetTiSA features allows for a single monitoring device to effectively detect a wide range of threats, from Botnets to Virtual Private Network (VPN), even in high-speed backbone infrastructures. Furthermore, the minimal telemetry overhead ensures that enhanced security does not come at the cost of excessive network bandwidth consumption.

Threat Detection using Flow Time Series Analysis

In this chapter, the threat detection using long-term periodic patterns is described. The novel method of detecting periodic behaviors was proposed in the thesis author's paper [A.4]. The Time Series Analysis (TSA) of Flow time series is successfully utilized for network traffic classification. However, the proposed method mines a limited feature vector, which cannot distinguish between similar classes in the same category, for example, different social networks. Moreover, the method requires a significant amount of resources, which complicates its deployment in real-world networks. Therefore, an enhanced method was proposed in the thesis author's paper [A.5]. The enhanced method simplifies detection, making it more feasible for deployment, and increases the feature vector by incorporating statistical-based and frequency-based features. The enhanced method was evaluated on the Cryptomining detection task [A.5] and Command and Control (C&C) detection task [A.6]. In both cases, the method achieved comparable results with State of the Art (SOTA). Moreover, the deployment of the enhanced method was further evaluated.

5.1 Motivation

In network traffic, repeated or periodic patterns are frequently observed, arising from the inherent behavior of specific applications, Application Programming Interfaces (APIs), or malicious threats. Consequently, analyzing long-term behaviors is essential for effective network traffic monitoring. However, traditional packet-based monitoring is often unfeasible in modern infrastructures due to the immense traffic volume and the prohibitive resource requirements necessary for capturing and analyzing large number of packets. Therefore, analysis of periodic patterns using packet-based analysis is unfeasible.

Flow-based monitoring [195] mitigates these resource constraints by utilizing only aggregated information and statistics regarding communication. Nevertheless, critical behavioral patterns are often lost during this process of aggregation. While numerous research

works [94, 95, 104, 132, 134, 144, 145, 153, 158, 181, 226, 241, 254] have proposed flow extensions to provide insights into packet-level behaviors, these remain unsuitable for long-term analysis. Such extensions are typically constrained by active and inactive timeouts that terminate flows within minutes, or by specific packet limits in extensions like Sequence of Packet Lengths and Times (SPLT).

As both standard packet-based and existing flow-based approaches are inadequate for analyzing long-term repeated or periodic patterns, the thesis author’s publications [A.4, A.5, A.6] address this gap through the use of Time Series Analysis (TSA) applied to *Flow time series*. This approach enables the identification of repeated or periodic patterns across multiple flow records belonging to the same generating process, estimated via *Network dependency*. This methodology simultaneously addresses the problem of excessive data volume and the loss of long-term behavioral patterns. The data volume is managed through flow-based aggregation, while the loss of behavioral patterns is mitigated by analyzing sequences across multiple flows.

5.2 Periodic Behaviors

Two distinct types of periodic behavior are identified within the *Flow time series* extracted from network traffic. These patterns provide critical insights into the underlying nature of the generating process.

The first type of behavior is defined by the periodic repetition of a consistent volume of data over highly stable time intervals. This periodic behavior is called *Clear*. Although the time intervals may exhibit minor fluctuations due to processing delays at the source or jitter within network components, the overall variance remains minimal. An example of this highly deterministic behavior is illustrated in the time series of the *Mirai* malware communication shown in Figure 5.1. As observed in the figure, the number of packets and bytes remains constant across the entire observed period, resulting in a predictable and horizontal pattern. Such behavior is characteristic of automated botnet activities, such as heartbeats or Command and Control (C&C) polling, where the communication overhead is fixed.

The second type of behavior involves data points that are distributed periodically in time, similar to the first type, but where the volume of data varies between individual transmissions. This variance is typically influenced by the dynamic nature of the message content or application state. This type of periodic behavior is generated by processes that communicate at regular intervals, such as updates, synchronization, or background telemetry. This periodic behavior is called *Sinusoidal*.

An example of this behavior is presented in Figure 5.2, which displays the time series of the social network *Facebook*. In this instance, while the packets are sent at relatively regular intervals, the byte counts exhibit noticeable fluctuations. These variations represent the difference between simple “keep-alive” messages and larger data exchanges, such as notification updates or state synchronization. Identifying these patterns allows for the

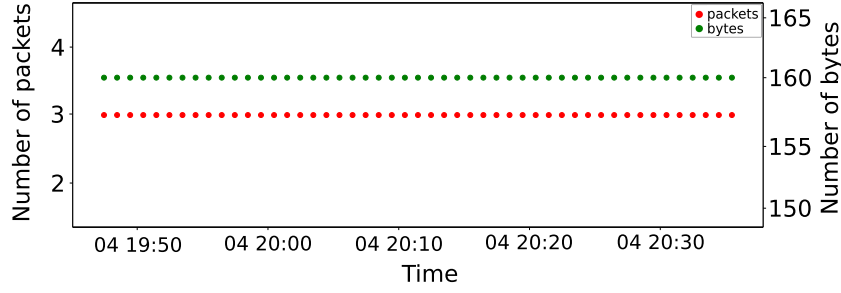


Figure 5.1: Periodic behavior of a time series of the malware Mirai. The plot demonstrates a highly stable pattern where both packet counts and byte volumes remain constant over time. This periodic behavior is called *Clear*.

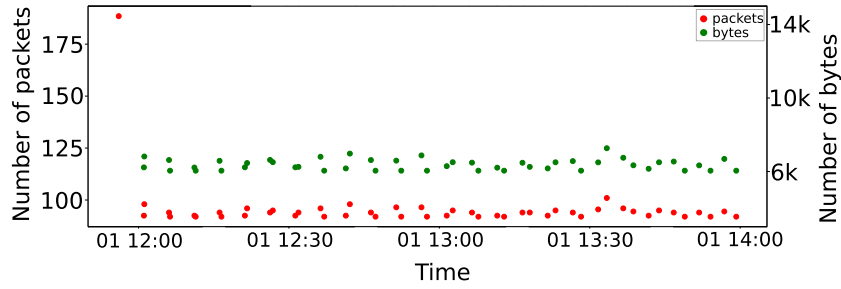


Figure 5.2: Periodic behavior of a time series of the social network Facebook. While the communication frequency remains periodic, the volume of bytes and number of packets fluctuate based on the specific content being synchronized. This periodic behavior is called *Sinusoidal*.

differentiation between strictly automated malicious scripts and more complex, yet still periodic, legitimate application background traffic.

5.3 Detection of Periodic Behaviors

To detect both types of periodic behaviors in network traffic, the usage of mathematical tools is necessary to detect periodic behaviors in the time series. The use of the engineering method, which would be possible for finding the first periodic behavior—constant periodicity—is not feasible due to noise in the data.

The analysis of unevenly sampled time series must compute not only the values of the data points but also their corresponding times. Unfortunately, most time series analysis methods specialize in evenly sampled time series, which are prevalent in many areas. Nevertheless, there are methods to analyse unevenly spaced time series.

The detection of periodic behavior proposed in the thesis author's paper [A.4] uses the *Lomb-Scargle periodogram*, which was defined by Lomb [141] and Scargle [203]. This periodogram can insert different sine signals into an unevenly sampled time series of periodic behavior and derive frequency and intensity for each, thus creating a periodogram.

The detection of periodic behavior proposed in the thesis author's paper [A.4] is pre-

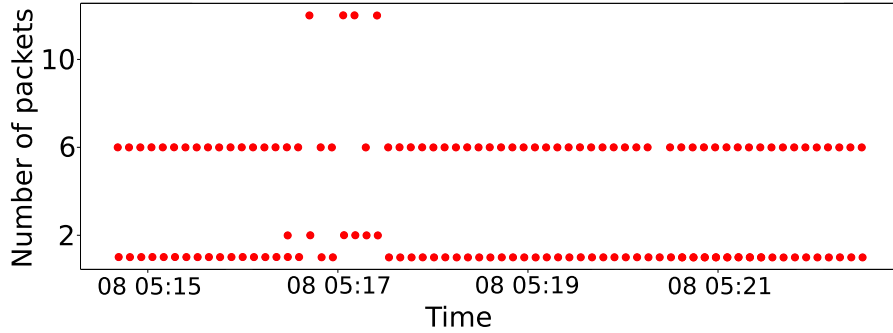


Figure 5.3: Example of *Flow time series* with *Network dependency* ff02::1:2(547)-fe80::1 representing communication with the DHCPv6 server. The presented periodic behavior is the periodic repetition of two flow records with numbers of packets equal to 1 and 6.

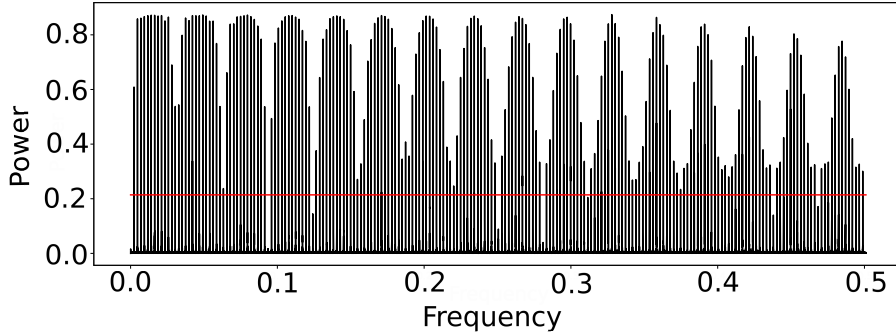


Figure 5.4: Lomb-Scargle periodogram applied on *Flow time series* (Figure 5.3). The red line at power equal approximately 0.2 is created by the test of significance. All peaks exceeding this line mark a statistically significant frequency of periodic behavior.

sented on the example of *Flow time series* shown in Figure 5.3. The overall detection pipeline will be shown on this example time series. The Lomb-Scargle periodogram applied on *Flow time series* (Figure 5.3) is shown in Figure 5.4.

The time series is periodic with period p , if there is a statistically significant peak at frequency $f = \frac{1}{p}$. It is therefore necessary to use a statistical significance test on the periodogram. The Scargle's Significance Test (SST) [70] is suitable for the Lomb-Scargle periodogram, which can be used to determine whether there is any statistically significant peak in the periodogram using a simple formula and also to find out whether a particular peak is statistically significant. The SST test of significance applied to the periodogram shown in Figure 5.4 is marked with the red line in the figure.

The SST uses the Cumulative Distribution Function (CDF) of Lomb-Scargle periodogram powers (P_{LS}), which is defined as probability $P[P_{LS} > z]$. As the observed power z becomes larger, it becomes exponentially less likely to be produced by pure noise. Then the observed power level is more likely due to an original deterministic (i.e., non-noise) feature in the time series [70].

The SST is performed for threshold value *Significance level*, which defines the tested

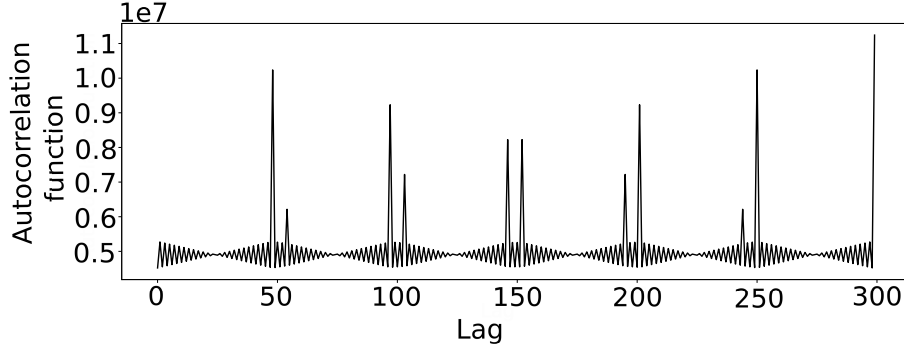


Figure 5.5: Autocorrelation function of the *Flow time series* (Figure 5.3) variable number of packets.

value:

$$z = \max(P_{LS}) * \text{Significance level} \quad (5.1)$$

The probability that z is bigger than rest of periodogram powers P_{LS} is computed by the equation:

$$P[P_{LS} > z] = 1 - e^{-\frac{z}{\sigma_{P_{LS}}^2}} \quad (5.2)$$

If $P[P_{LS} > z]$ is larger than or equal to the threshold value *Percentage level*, then the time series contains periodic behavior. The smaller the *Significance level* is, the stricter the periodic detection. Moreover, the bigger *Percentage level* is, the stricter the periodic detection.

However, in the periodogram, there are usually many frequencies with peaks that represent the same periodic behavior with a larger period. For example, the period $p = 2$ which causes a peak at frequency $f = \frac{1}{2}$, also causes peak at frequency $f = \frac{1}{4}$, $f = \frac{1}{6}$, $f = \frac{1}{8}$, and so on. Nevertheless, the other frequencies have different power on the periodogram.

First, the candidate p must be decided in order to confirm the periodicity of it using *Scargle's CDF* test applied on corresponding frequency $f = \frac{1}{p}$. The candidate is selected based on an autocorrelation function. However, the autocorrelation function does not account for time in its processing, so it is applied only to the values in the series, and time is neglected. An example of applying the autocorrelation function to the variable number of packets of the *Flow time series* (Figure 5.3) is shown in Figure 5.5.

The candidates on periodicity are selected based on the distances between the peaks of the autocorrelation function. The histograms of distances between the autocorrelation function's peaks are created for each variable of the *Flow time series* (i.e., number of packets, number of bytes, time difference between intermediate data points of time series). The histograms for each variable for the *Flow time series* (Figure 5.3) are shown in Figure 5.6.

The result is a set of candidates for periodicity that significantly appear in all histograms, which can be verified subsequently using the Lomb-Scargle periodogram and the *Scargle's CDF* test. In the example shown in Figure 5.6, period $p = 2$ is chosen as the

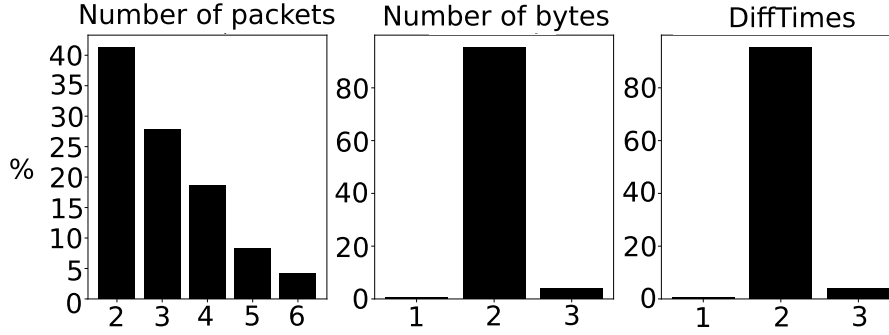


Figure 5.6: Histograms of distances between peaks of the autocorrelation functions applied on variables of the *Flow time series* (Figure 5.3).

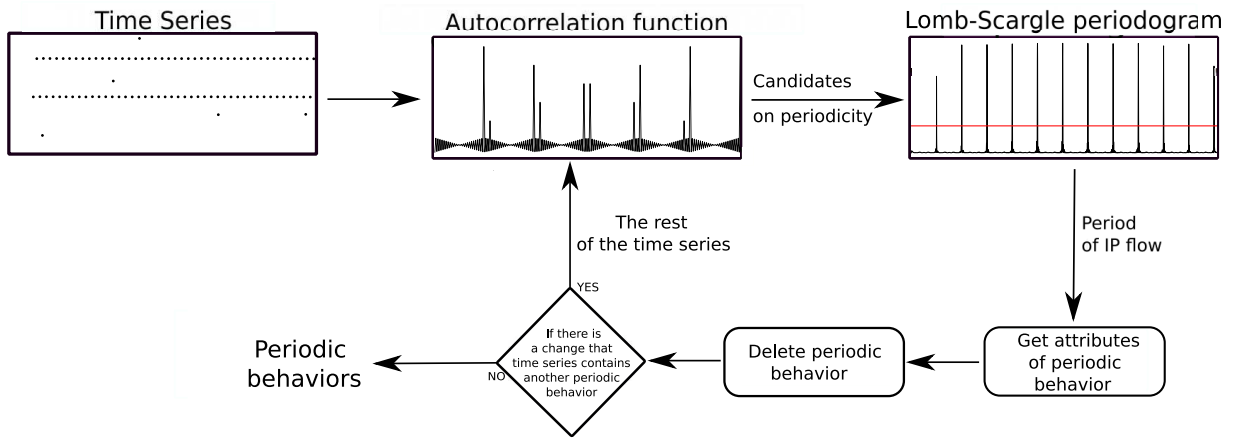


Figure 5.7: Methodology of periodicity detection approach proposed in the thesis author's paper [A.4].

candidate, and a statistically significant peak is then found in the vicinity of the frequency $f = \frac{1}{2}$ in Lomb-Scargle periodogram (Figure 5.4). Thus, the period is confirmed in the example.

A natural description of periodic behavior from Figure 5.1 is *Number of flow records* that periodically repeat, *Time period*, *Number of packets* which repeat, and *Number of bytes* which repeat. However, these attributes cannot perfectly describe the different periodic behavior in Figure 5.2 due to its higher variability and noise. Therefore, *Boundaries of the interval* of the data points are used for this type of periodic behavior. The boundaries embody most of the data points of periodic behavior for each time series variable.

In general, *Network dependencies* may not always split traffic into ideal *Flow time series*, as described before; it is the estimation that could fail. Therefore, a single *Flow time series* may exhibit multiple periodic behaviors. Therefore, in the methodology, after successfully detecting periodic behavior, a check is made to determine if the time series contains other periodic behavior. The diagram of the whole method is shown in Figure 5.7.

5.4 Classification of Applications, Services, and Operating Systems

In the thesis author's paper [A.4], the usability of the proposed periodicity detection approach was evaluated in a multiclass classification task focused on recognizing network traffic.

Dataset

The dataset of attributes for periodic behaviors was created to evaluate the possibility of using the proposed approach for classifying applications, services, and operating systems. The dataset contains 26 thousands samples divided into 61 classification classes. As a traffic source for the dataset creation, multiple publicly available datasets were used: [67], [217], [50], and [68]. Additionally, the dataset also contains the traffic of CESNET2, the traffic of the Network monitoring laboratory at FIT, the Czech Technical University in Prague, several home networks, and the communication of Android mobile devices.

The classification classes of the dataset contains following categories of traffic: social networks (e.g., Facebook, MS teams, Slack), remote storage (e.g., Google Drive, OneDrive, Github), updates of operating systems (e.g., windows, ubuntu), antivirus programs (e.g., Eset, Avast, Kaspersky), game clients (e.g., Steam, Epic Games, Uplay), network services and protocols (e.g., Keep-alive, HTTP2 ping, DNS), email browser viewers and clients (e.g., Gmail, Outlook), multimedia streaming (e.g., youtube, itunes, spotify), and web browsing software (e.g., firefox, opera).

Evaluation Methodology

The features of periodic behaviors inside created dataset were evaluated by creating a novel classifier for a multiclass classification task. The classifier creation pipeline is the set of steps that creates the best final model. Initially, the dataset is split into training and testing sets using a 70:30 split, while maintaining a similar labeling ratio in both sets.

In the validation phase, the optimal hyperparameters of Machine Learning (ML) algorithms were selected using a validation set split from the training set. The set of evaluated ML algorithms is shown in Table 5.1.

Experimental Evaluation

The effectiveness of the proposed periodicity detection was demonstrated through a series of multiclass classification experiments. The results revealed a significant performance distinction between the two types of periodic behaviors previously identified.

Traffic characterized by the second type of periodicity, which includes social network communications such as *Facebook*, *Discord*, and *TikTok*, was found to be more challenging

Table 5.1: Results of machine learning algorithms in %.

Model	Accuracy	Precision	Recall	F1-score
<i>Naive Bayes</i>	8	25	8	4
<i>Logistic Regression</i>	42	19	42	26
<i>kNN</i>	62	62	62	61
<i>Extra Tree</i>	65	66	66	66
<i>Decision Tree</i>	77	77	77	77
<i>Random Forest</i>	89	90	89	88
<i>XGBoost</i>	91	90	90	90

to classify accurately. As illustrated in the confusion matrix in Figure 5.8, substantial misclassification occurs among social platforms. For instance, while *Facebook* is recognized with high confidence (0.96), other platforms like *LinkedIn*, *Reddit*, and *Twitter* show significant confusion with the *Facebook* class, likely due to similarities in background polling and synchronization patterns.

Nevertheless, system and application-level traffic—such as *keep-alive* signals, *HTTP2 ping* messages, and Domain Name System (DNS) queries—align with the first type of periodicity. These communications exhibit highly stable patterns, as shown in the example of the *Mirai* malware (5.1). In these cases, the classifier is able to recognize each class with significantly higher accuracy. As seen in Figure 5.9, most services and protocols achieve near-perfect classification scores (often reaching 0.99). Furthermore, it was observed that accuracy increases proportionally with communication frequency; more frequent background polling provides a denser set of periodic features for the model to analyze.

The evaluation of various machine learning algorithms, summarized in Table 5.1, highlights the superiority of ensemble methods. While baseline models like *Naive Bayes* and *Logistic Regression* performed poorly on the complex multiclass dataset, tree-based models provided robust results. The *XGBoost* algorithm achieved the highest overall performance with an accuracy of 91% and an F1-score of 90%, closely followed by *Random Forest* at 89% accuracy.

5.5 Enhanced Detection of Periodic Behaviors

The method for the detection of periodic behavior from the thesis author’s paper [A.4] is not feasible for high-speed Internet Service Provider (ISP) networks because it can process only a small number of time series per second. Therefore, optimizations would be required for deployment. Moreover, the attributes obtained by the approach cannot distinguish between similar classes in the same category (for example, different social networks).

In the thesis author’s paper [A.5], an enhanced detection approach was proposed. A novel and faster periodicity detection pipeline presented in Figure 5.10 is designed for

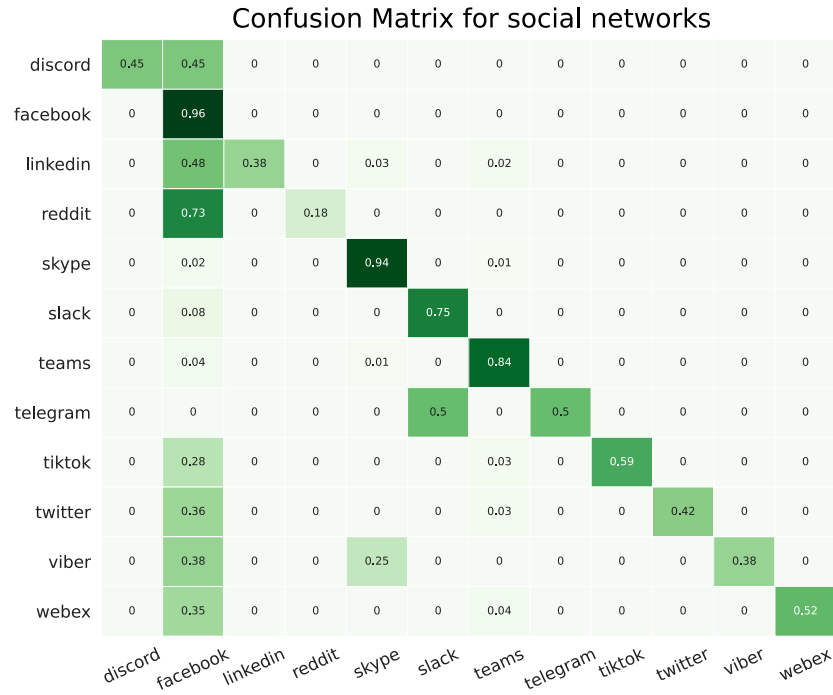


Figure 5.8: The confusion matrix for social networks

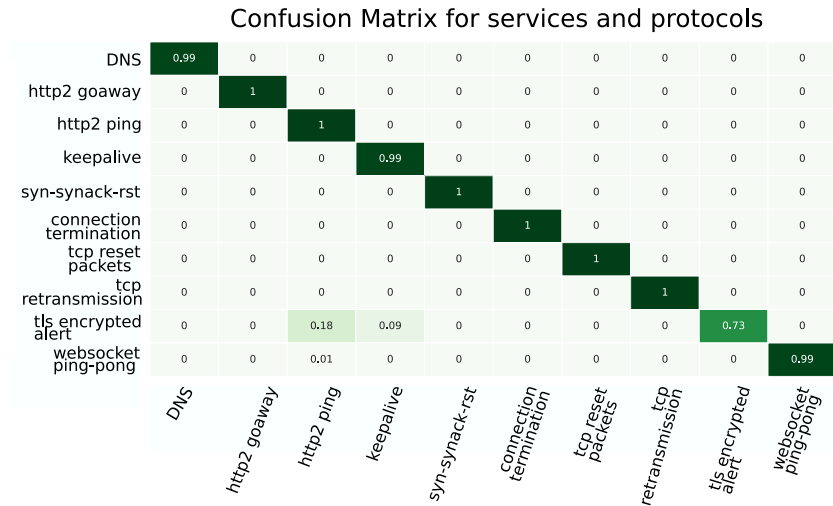


Figure 5.9: The confusion matrix for services and protocols

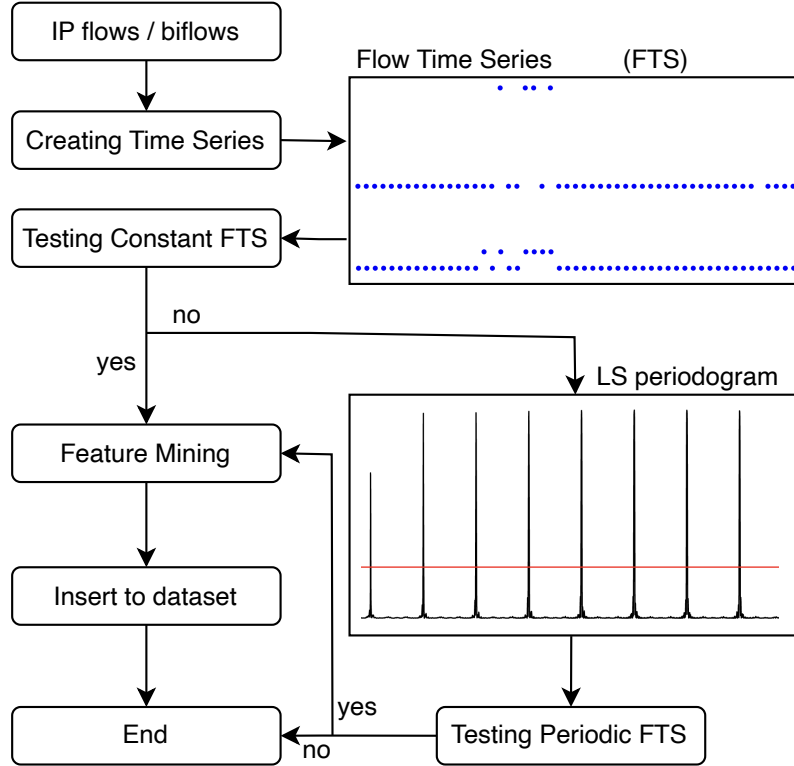


Figure 5.10: Methodology of enhanced periodicity detection approach proposed in the thesis author’s paper [A.5].

high-speed networks. This pipeline is able to process the whole CESNET-MINER22 [180] dataset (containing 3 million flows obtained during 3 months) in 5 seconds with proper precision based on two threshold values of SST.

The previous approach could fail for constant *Flow time series*, i.e., the same datapoint occurs periodically. Because the creation of the Lomb-Scargle periodogram results in failure. Moreover, the constant time series can be easily tested using the histogram approach, thereby speeding up the detection process. Therefore, an enhanced detection approach tests if the *Flow time series* is constant and mark such time series as periodic.

Moreover, the periodic behavior detection pipeline can cause false positives for short *Flow time series*. Therefore, the detection pipeline is run only for the *Flow time series* with at least 10 datapoints (i.e., flows).

Feature mining

The *Flow time series* in this method have two time series metrics: the number of packets and bytes within the IP flow. Furthermore, each datapoint has two time-related pieces

of information: the time of transmission of the first packet (t_{f_i}) and the last packet (t_{l_i}). Moreover, from the time information, two more time series metrics can be created: the durations and time differences.

The duration for each flow is computed as $D = t_{l_i} - t_{f_i}$, forming a time series metric. The time differences are calculated for neighboring flows as $Dt = t_{f_i} - t_{l_{i-1}}$. These time series metrics and the Lomb-Scargle periodogram are input data for the computation of features for ML.

In total, the enhanced approach presents 43 features which are grouped into two groups to describe both *Clear* and *Sinusoidal* periodic behaviors. The first group of *Clear* periodic behavior features contains: *Number of packets*, *Number of bytes*, *Duration*, and *Time difference*. The second group of *Sinusoidal* periodic behavior features contains pairs of features as borders of interval where *Sinusoidal* periodic behavior occurs. Thus, there is pair of features representing interval for each features for *Clear* periodic behavior.

Moreover, the feature set also includes basic statistical properties (*Mean*, *Standard Deviation*, *Skewness*, and *Kurtosis*) for each time series metric, resulting in 16 statistically based features. Furthermore, the Lomb-Scargle periodogram is used to generate the set of well-known frequency-based features such as *Min power*, *Frequency of min power*, *Spectral bandwidth*, *Spectral centroid*, *Spectral energy*, and *Spectral entropy*.

5.6 Detection of Cryptomining

In the thesis author’s paper [A.5], the enhanced approach is proposed and evaluated on the Cryptomining detection task. Cryptocurrencies first appeared in 2008, when Satoshi Nakamoto published Bitcoin’s whitepaper [166]. Since then, many cryptocurrencies have been created. A decentralized system of money, where transactions are verified by individuals known as miners and mutual trust is established through cryptography, has quickly become very popular.

Many cryptocurrencies are based on Proof-of-Work (PoW) mechanisms, which consume a lot of electricity and processing power. Today, Proof-of-Stake (PoS) mechanisms, which do not require demanding computations, are becoming popular — Ethereum switched to the PoS mechanism in 2022¹. However, PoW is still used dominantly². Miners use electricity and processing power to verify transactions (mining), which is rewarded in the form of crypto coins³.

However, mining can be easily exploited by attackers to gain money and therefore remains in high positions on lists of cybersecurity threats. The Cisco Umbrella Academy Report from 2021⁴ revealed that 69% of organizations encountered some level of unsolicited cryptomining. Attackers use “abusive cryptomining” to mine cryptocurrencies on

¹<https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/>

²<https://www.forbes.com/advisor/investing/cryptocurrency/proof-of-work/>

³<https://developer.bitcoin.org/devguide/mining.html>

⁴<https://learn-cloudsecurity.cisco.com/umbrella-library/2021-cyber-security-threat-trends-phishing-crypto-top-the-list>

the victims' computers. As a result, attackers receive new cryptocurrency coins for free, while victims incur high electricity bills and sometimes even damaged devices. Furthermore, *Cryptojacking* [91] is a technique in which mining code (written in JavaScript) is embedded into a website, and all who visit such a website will become victims of abusive mining. The second way to perform abusive mining is directly through *binary*-based mining malware [176]. Such binaries can be delivered in a phishing email or as part of a regularly downloaded file.

Abusive cryptomining remains an active and commonly employed attack. Therefore, highly accurate and effective detection methods are still necessary to protect users worldwide. Existing detection methods utilize ML with basic statistical features derived from flow-based network telemetry. Nevertheless, actions performed by cryptocurrency miners repeat periodically, and time-related features such as periodicity are essential in the network classification [158]. At the time of writing the thesis author's paper [A.5], a method to detect cryptocurrency miners based on their periodic behavior was missing.

Creating Datasets

In the evaluation of cryptomining detection, experiments were conducted with varying lengths of the *Flow time series* by defining the time interval during which the *Flow time series* are collected. The chosen time intervals are: 24 h, 12 h, 6 h, 4 h, 2 h, 1 h, 30 m, 15 m, and 10 m. The IP flows were taken from the CESNET-MINER22 dataset, which was created by monitoring the CESNET2 network infrastructure by ipfixprobe — open-source flow exporter. The computed *Flow time series* created from the CESNET-MINER22 [180] dataset and the datasets of periodic behaviors have been published on Zenodo [A.19].

The number of created *Flow time series* depends on the size of the time interval. In the top left graph of Figure 5.11, the number of *Flow time series* increases with the decreasing time interval. The number of constant *Flow time series* achieved the same trend as can be seen in the bottom left graph of Figure 5.11. However, the top right graph of Figure 5.11 shows the number of tested *Flow time series* with at least ten datapoints, and it is not increasing with the same trend.

The number of periodic *Flow time series* depends on the settings of SST, i.e., *Significance level* and *Percentage level*. The right bottom graph of Figure 5.11 shows the numbers of periodic *Flow time series* for each tested setting. The number of periodic *Flow time series* with *Significance level* equal to 0.1 has the same trend as the number of tested *Flow time series*. The number of periodic *Flow time series* with *Significance level* equal to 0.01 has a decreasing trend with decreasing time intervals. The number of periodic *Flow time series* with a *Significance level* of 0.001 exhibits a slightly increasing trend with decreasing time intervals.

Based on the premise that the longer the time interval, the more accurate the periodicity test should be, and based on the described observation, the correct SST setting should be *Significance level* equal to 0.01. Nevertheless, the experiments were performed with the settings of SST as a hyperparameter in classification to obtain the best classification results.

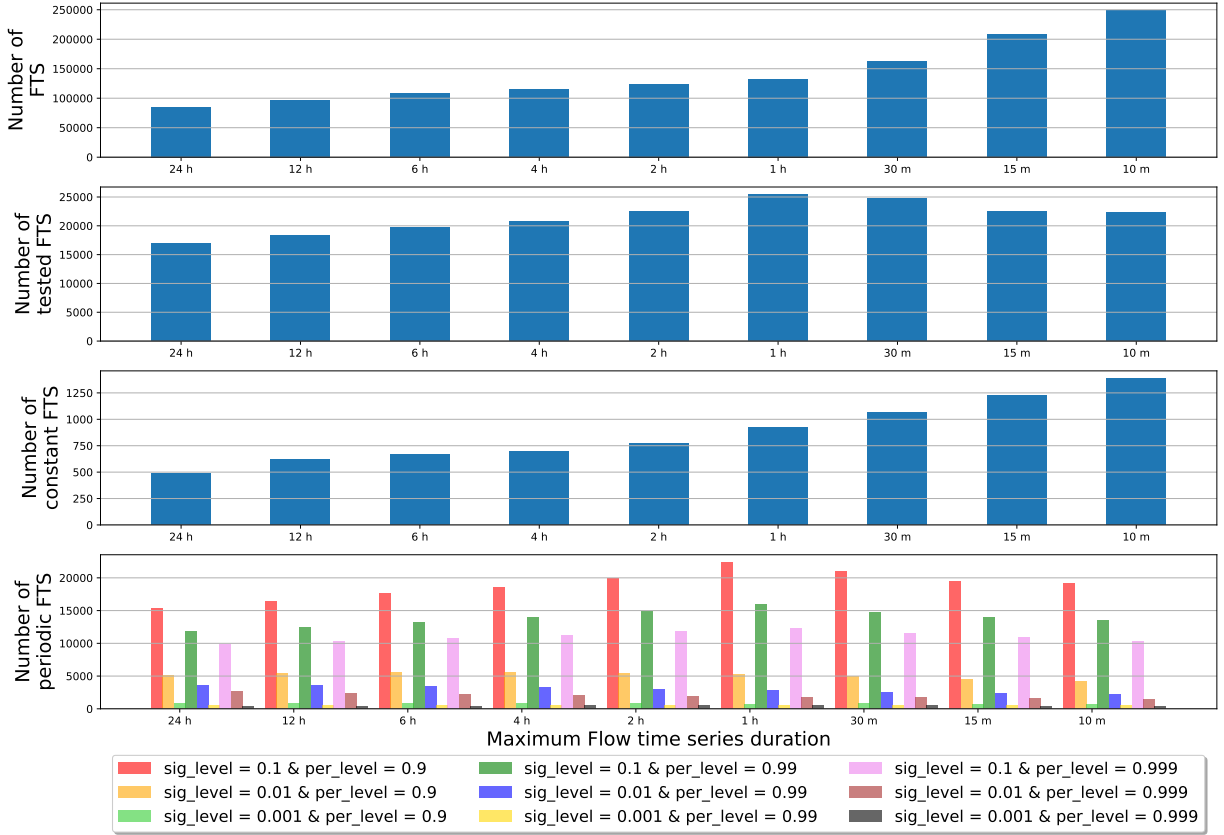


Figure 5.11: The figure presents the number of *Flow time series*, the number of tested *Flow time series*, and the number of constant *Flow time series* and periodic *Flow time series* in datasets of periodic behavior created from the CESNET-MINER22 dataset. The number of periodic *Flow time series* depends on SST settings.

The classification pipeline uses only flows from periodic *Flow time series*. The number of classified flows from non-periodic *Flow time series* is shown in Figure 5.12. This figure shows results with SST *Significance level* set to 0.01 and *Percentage level* set to 0.99. The figure also shows that the number of flows inside non-periodic *Flow time series* increases with smaller time intervals. Furthermore, the non-periodic *Flow time series* usually contain a small count of datapoints, only one in most cases.

Evaluation Methodology

The evaluation methodology is a set of steps that lead to the creation of the best final model. Firstly, Plný et al. [180] split the CESNET-MINER22 dataset into the *Design* and *Evaluation* parts. The *Design* part is for creating, tuning, and selecting the best model, and the *Evaluation* part is for the one-time evaluation of the best model from the *Design* part. This default split avoids overfitting and enables the comparison of results with other works without variation due to random sampling.

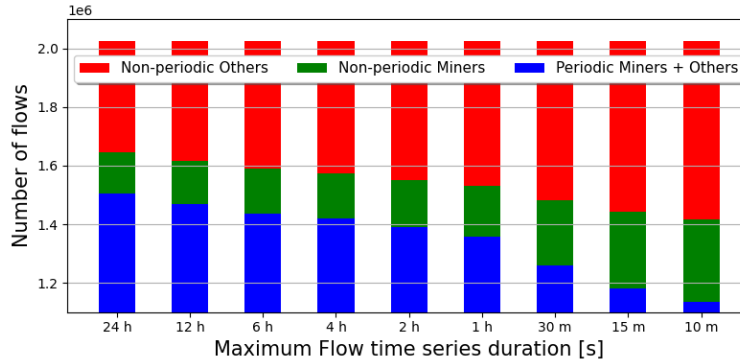


Figure 5.12: The number of flows that are in periodic and non-periodic *Flow time series*.

In the design phase, the optimal ML algorithm was selected by testing 11 well-known ML algorithms, from which XGBoost, LightGBM, and CatBoost achieved similar results. The XGBoost algorithm was chosen. The experiments ended similarly on each dataset (different time interval).

After selecting the ML algorithm, the hyperparameters of the XGBoost model were optimized to obtain a suitable set that performs well on the dataset while avoiding overfitting. The *hyperopt library* [29] was used to tune the following hyperparameters: *n_estimators*, *max_depth*, *gamma*, *reg_alpha*, *reg_lambda*, *min_child_weight*, and *colsample_bytree*. This tuning phase yields the optimal settings for the hyperparameters of the XGBoost classifier. In the design phase, the best models were picked for each dataset. The best models for each time interval were selected based on the evaluation metrics: *Accuracy*, *Precision*, *Recall*, and *F1-score*.

After the design phase, the selected models were evaluated on the *Evaluation* part of the CESNET-MINER22 dataset. The source code of the classification pipeline and experiments is published on GitHub⁵.

Experimental Evaluation

The best results for settings *Significance level* equal to 0.1 and *Percentage level* equal to 0.9 are presented in Table 5.2. As can be seen, the periodic features are more than suitable for detecting cryptocurrency miners.

The results of the feature importance investigation for the trained model are shown in Figure 5.13. The *Clear* periodic behavior features are one of the least important, but they cause a low false-positive rate, as found out in the previous thesis author's paper [A.4]. The most important features are more suited for *Seasonal* periodic behavior, which indicates that recognition between benign and cryptomining periodic communication is between *Seasonal* periodic behaviors.

For deployment into networks for real-time detection, it is crucial to choose a proper time interval for *Flow time series* sampling and to set the proper settings of the SST test.

⁵<https://github.com/koumajos/EnhancedDeCrypto>

Table 5.2: Summary of best results with settings *Significance level* equal to 0.1 and *Percentage level* equal to 0.9.

Time interval	Accuracy	Precision	Recall	F1-score
24 h	93.63	83.49	74.19	78.57
12 h	93.31	87.82	69.27	77.45
6 h	93.37	90.01	74.87	81.74
4 h	94.19	92.24	78.88	85.04
2 h	94.34	93.44	80.46	86.47
1 h	94.86	94.21	84.39	89.03
30 m	95.13	96.11	87.45	91.57
15 m	95.19	95.43	89.21	92.21
10 m	94.92	95.16	88.88	91.91

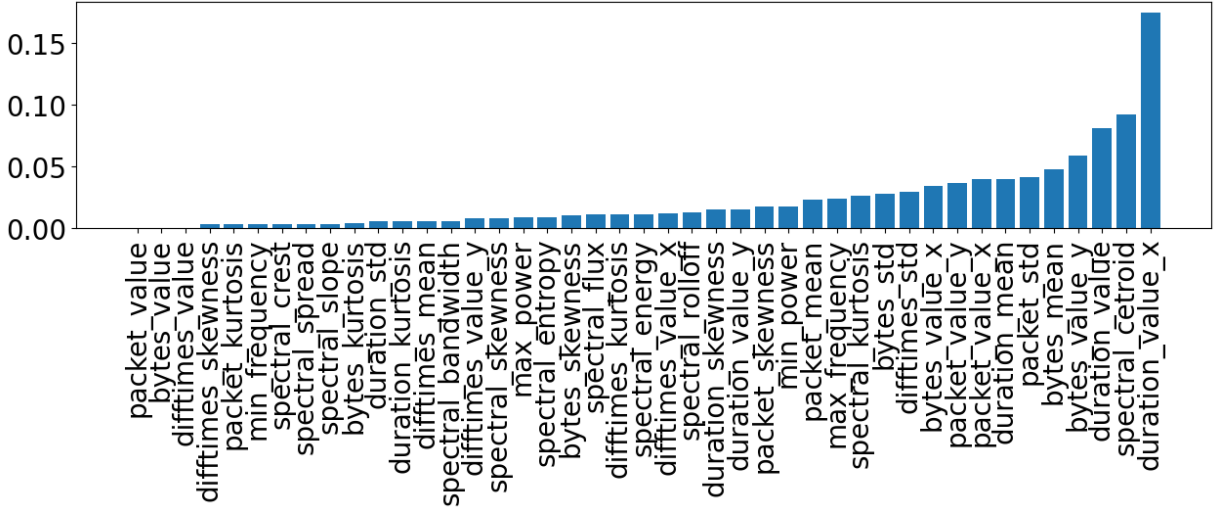


Figure 5.13: Feature importance

5. THREAT DETECTION USING FLOW TIME SERIES ANALYSIS

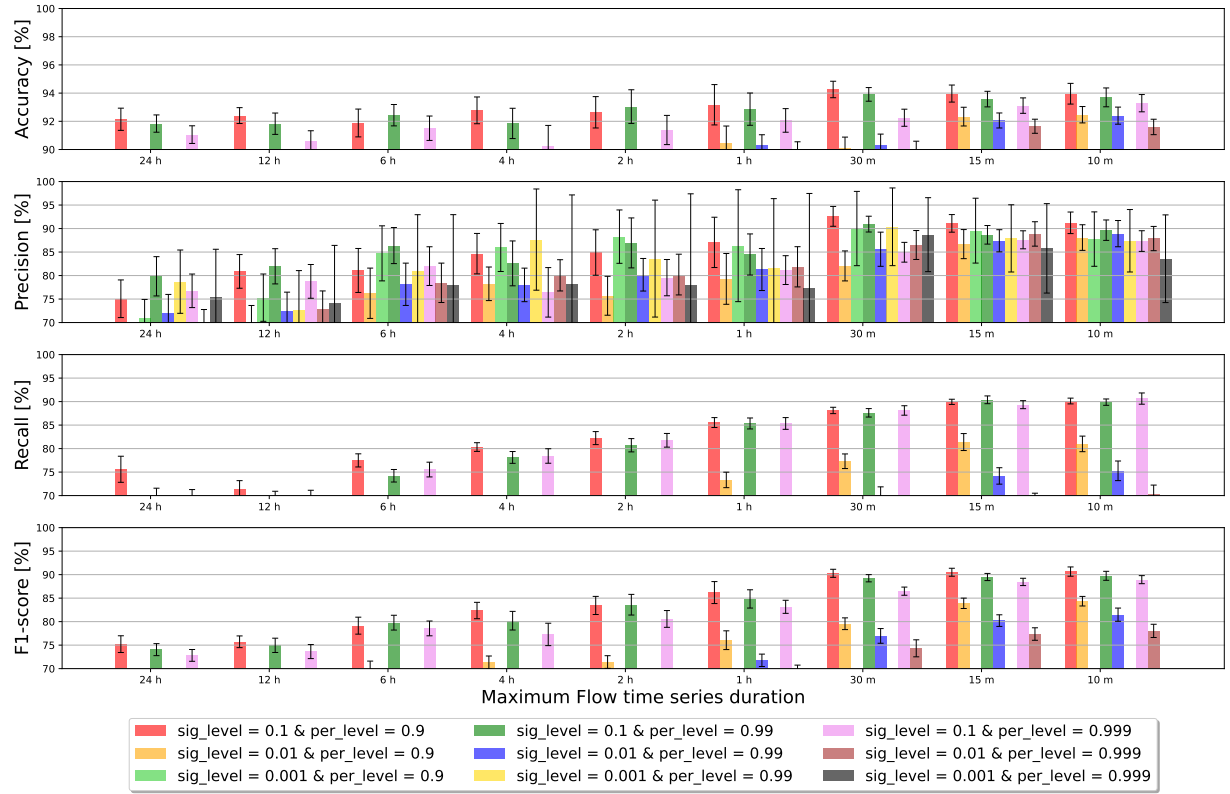


Figure 5.14: The graph contains results of the ML pipeline for each dataset that was run 100 times to obtain significant statistical results. Therefore, the presented results are the mean of the classification metric with standard deviation. On the graph, the dependency between ML results, the time window of creation time series, and the settings for periodicity detection can be seen.

However, the best achieved results, presented in Table 5.2, can not be used for selecting a proper time interval because most of them achieved similar results. Moreover, the results can be strongly influenced by variation caused by the randomness that naturally occurs in splitting, training, and tuning of ML models [10]. Therefore, the classification pipeline for each time interval was evaluated 100 times. The results of this experiment enabled the selection of the correct time interval for deployment. The resulting mean and standard deviation of each classification metric, for each time interval and SST setting, is shown in Figure 5.14.

The experiments show that most of the best results are caused by variation, as the mean values of each classification metric do not achieve the same or nearly the same value. However, the mean and standard deviation helped to select the time interval equal to 30 m, *Significance level* equal to 0.1 and *Percentage level* equal to 0.9. This setting occurs with the best average results and the smallest standard deviation; thus, the results are robust.

5.7 Detection of Botnet's C&C Communication

In the thesis author's paper [A.6], the enhanced approach is evaluated on the botnet's C&C communication detection task. C&C is the communication channel usually established for malicious activities through backdoors or remote shells. The C&C allows attackers to maintain persistent access to compromised systems, issue commands, and exfiltrate data. Detection of C&C communication in network traffic has become essential for maintaining the security of enterprises [148]. Most recent threats, including Advanced Persistent Threat (APT) [156], employ some form of C&C. APT groups consistently employ C&C techniques to maintain persistent access to compromised systems, exfiltrate data, and coordinate further malicious activities [150]. Analysis of documented APT groups in the MITRE ATT&CK framework⁶ reveals that all such groups rely on at least one C&C mechanism, ranging from standard protocols (e.g., Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS), DNS) to more covert channels such as custom encrypted tunnels or legitimate services like cloud storage and social media platforms [136].

Detecting botnet activity on individual devices using antivirus or Endpoint Detection and Response (EDR) systems is not feasible in environments where direct access to endpoint devices is not possible. This challenge is prevalent among ISPs, organizations managing large-scale networks, and operators of public WiFi infrastructures. Nevertheless, botnet detection in network traffic is challenging due to the high adoption of encryption to increase the privacy of users on the Internet [177].

Relying solely on blacklists of known C&C server IP addresses is inadequate, as botnets frequently alter their infrastructure within days [233]. Therefore, the adoption of ML is a viable option in both packet-based and flow-based detection [238]. In the packet-based detection, multiple approaches were evaluated, for example, a small set of packet-level features with an explainable gradient-boosted tree model [12], and a semi-supervised One-Class Support Vector Machines (SVM) trained on packet-header features [124], and aggregated packet time series per 1s interval with a K-Nearest Neighbors (k-NN) model [100]. In the flow-based detection, multiple features and models were adopted, for example, time-series statistics and XGBoost model [A.2], graph-based features with self-organizing map model [47], transfer learning approach by clustering of tabular features [107], multi-modal features with random forest model [185], and sequence of packet lengths with deep learning models [255].

However, more resilient botnets, particularly those leveraging advanced architectures [171], further complicate detection efforts. Despite sophisticated evasion techniques, such as encryption, protocol obfuscation, and random padding, the periodic nature of botnet communication remains a detectable trait [21, 76, 274]. Therefore, the thesis author's approach focuses on identifying periodic communication patterns, a fundamental characteristic of botnet activity.

⁶<https://attack.mitre.org/>

Datasets

The detection approach based on the periodic nature of botnet communications was evaluated using two datasets. The first is the widely recognized CTU-13 dataset [72], which has become a standard benchmark in botnet detection research due to its extensive collection of real-world network traffic with labelled botnet activities. Despite its widespread use, CTU-13 is over a decade old, and its relevance to current cybersecurity threats is limited. Over the years, botnet architectures, C&C techniques, and evasion strategies have evolved significantly [238], making the dataset less representative of modern botnet behaviors. Additionally, most scenarios in CTU-13 have a short time length, which restricts their suitability for evaluating multi-flow detection methods to uncover persistent C&C communication.

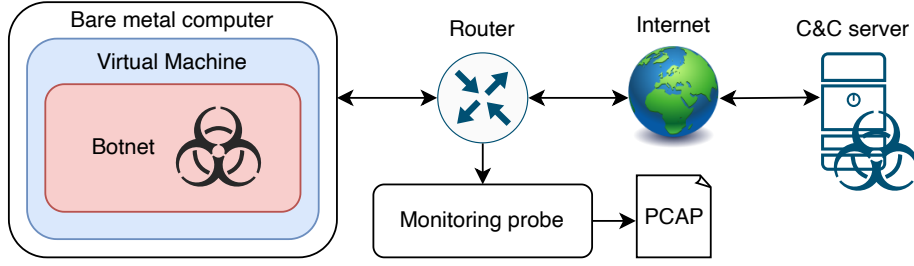
To overcome these limitations, a novel dataset, CESNET-CC25, is introduced, specifically designed to reflect contemporary botnet C&C traffic patterns. CESNET-CC25 offers a richer and more representative set of real-world threats, capturing modern behaviours such as decentralized architectures and encrypted communication channels. This enables more accurate and robust evaluation of detection methods under realistic and current conditions. To promote reproducibility and encourage further research in this domain, CESNET-CC25 is publicly available via the Zenodo platform [A.23, A.24, A.25].

Capturing Process of CESNET-CC25 Dataset

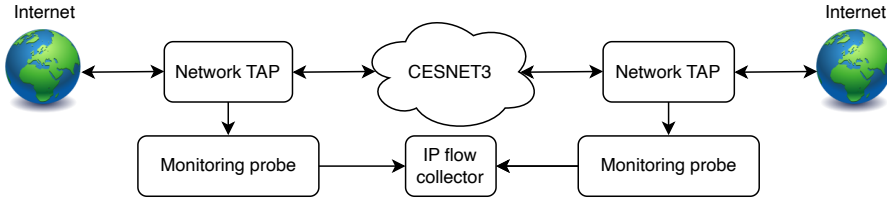
Capturing and distinguishing botnet communications in real-world network environments poses a significant challenge, as C&C traffic typically constitutes only a negligible fraction of the total network flow. Nevertheless, capturing realistic benign traffic in a controlled laboratory setting is equally problematic, as such environments cannot faithfully replicate the diversity and scale of real-world communication patterns. To address these limitations, two distinct traffic capture architectures were designed and deployed, enabling the collection of authentic C&C traffic from active botnets and representative benign traffic from operational networks.

The first architecture is dedicated to capturing botnet C&C communication, as illustrated in Figure 5.15a. In this setup, botnet binaries were executed within isolated virtual machines. Upon activation, most botnets immediately initiated communication with their respective C&C servers. This traffic was intercepted at the router level using port mirroring and recorded with tcpdump on a dedicated monitoring probe. Continuous traffic supervision ensured timely intervention, and all malicious activities were promptly identified and blocked. For instance, the Gafgyt botnet typically initiates attack traffic within seconds of execution.

The second architecture was designed to collect benign traffic from the CESNET3 network, as depicted in Figure 5.15b. This infrastructure, deployed within a live ISP network, enabled us to obtain real-world benign communication patterns. Such an approach provides a more authentic representation of legitimate traffic than synthetically generated traffic in laboratory settings, which often fails to capture the complexity and variability of



(a) Capture architecture of botnet traffic in laboratory environment



(b) Capture architecture of benign traffic from ISP network CESNET3

Figure 5.15: Capturing process of the CESNET-CC25 dataset

real network behaviour. Importantly, it also addresses the standard limitation of existing malware communication datasets, which often include laboratory-created benign traffic.

Content of CESNET-CC25 Dataset

The CESNET-CC25 dataset contains traffic from 32 distinct botnet binaries representing five different botnet families that were among the most prevalent at the time of writing. Each capture, along with associated traffic statistics, is summarized in Table 5.3. The raw botnet communication traffic is provided in PCAP format and publicly available on Zenodo [A.23] for detailed analysis.

Furthermore, the CESNET-CC25 dataset is available in two formats—IP flows extracted by the `ipfixprobe` exporter, available on Zenodo [A.24], and time series features distilled for the purpose of the periodicity detection approach⁷, available on Zenodo [A.25], both in CSV format.

Analysis of CESNET-CC25 Dataset

The dataset consists of IP flows representing both benign and botnet communications, as previously described. These IP flows serve as the fundamental layer upon which time series features are subsequently extracted, and the detection method is built.

To support reproducible evaluation, the dataset is already split chronologically into training and testing subsets, following an approximate 70:30 ratio of time. Detailed statistics on the number of flows in each split are presented in Table 5.4.

⁷<https://github.com/koumajos/EnhancedDeCrypto>

Table 5.3: Botnet captures inside CESNET-CC25 dataset

No.	Family	Dur. [h]	C&C packets	C&C IP flows
1	Gafgyt	138.21	33 793	8 290
2	Mirai	24.71	92 346	22 263
3	Mirai	25.43	10 767	1 954
4	Mirai	25.23	6 256	1 532
5	Mirai	25.07	10 527	1 879
6	Kaiji	167.01	145 787	3 635
7	Mirai	20.43	121 308	17 464
8	Mirai	20.17	116 199	16 609
9	Miori	8.60	19 889	4 938
10	Hailbot	54.57	16 024	3 946
11	Gafgyt	3.12	3 462	975
12	Mirai	1.98	548	126
13	Hailbot	22.02	4 300	1 056
14	Hailbot	35.25	6 851	1 663
15	Mirai	47.32	50 138	583
16	Gafgyt	10.39	21 368	125
17	Hailbot	70.02	10 858	2 540
18	Mirai	111.44	28 078	6 970
19	Mirai	68.98	14 695	3 412
20	Gafgyt	15.41	3 738	928
21	Mirai	22.49	77 691	18 918
22	Gafgyt	42.94	10 640	2 610
23	Gafgyt	4.05	987	245
24	Mirai	150.22	36 415	9 040
25	Xorbot	43.39	156 215	2 646
26	Gafgyt	14.51	9 892	2 826
27	Hailbot	2.03	338	94
28	Gafgyt	42.49	53 901	11 089
29	Mirai	103.99	26 103	6 291
30	Gafgyt	1.23	298	75
31	Gafgyt	30.41	7 697	1 825
32	Mirai	27.34	15 634	4 196
Total		1 380.44	996 743	160 743

Table 5.4: Standardized train and test split of IP flows in the dataset.

Traffic	Train IP Flows	Test IP Flows	Total
Benign	132683119	61467627	194150746
Botnet	97120	63623	160 743

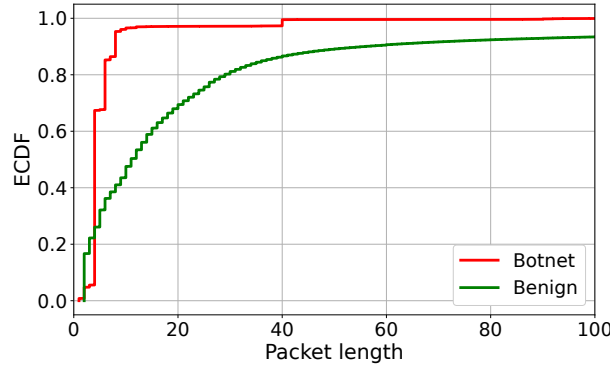


Figure 5.16: Distribution of packet lengths of IP flows of benign and botnet communication.

The majority of botnet flows in the dataset are short, with 95% containing up to 10 packets. These short flows primarily represent repeated establishment of connections with C&C servers. In contrast, a noticeable spike in longer flows around 40 packets indicates connections where some data was exchanged, for example, an attack order. Overall, botnet traffic exhibits relatively uniform patterns across the dataset, whereas benign traffic is significantly more diverse in its characteristics, as illustrated in Figure 5.16.

When examining individual IP flows and their characteristics, botnet and benign traffic can often appear indistinguishable due to low variability in their observable features. As demonstrated by Jeřábek et al. [105], packet sequence metadata—commonly used in traffic classification—tends to exhibit a high degree of duplication. This observation holds true in CESNET-CC25 dataset as well. The botnet portion, comprising 160,745 flows, contains only 2,458 unique packet length and direction sequences. In contrast, the benign traffic is substantially more diverse, with 54,987,920 unique sequences.

Furthermore, these sequences are from some part shared between botnet and benign classes, making direct classification based on a single IP flow unreliable. This overlap further motivates the need for alternative detection strategies, such as the approach based on *Flow time series* and periodicity.

Evaluation Methodology

The approach focuses on detecting periodic communication patterns, a core behavioural trait of many botnets. The overall methodology is illustrated in Figure 5.17. After packet capture, traffic is aggregated into IP flows as described in earlier sections. For the CTU-13 dataset, the flow representations are already prepared by the original authors.

The IP flows are chronologically divided into training and testing sets using a 70:30 split along the time axis. Subsequently, flows are segmented into fixed-length time windows of duration t —a key hyperparameter of the method. Within each time window, IP flows are transformed into multiflow time series, each corresponding to a specific *Network dependency*.

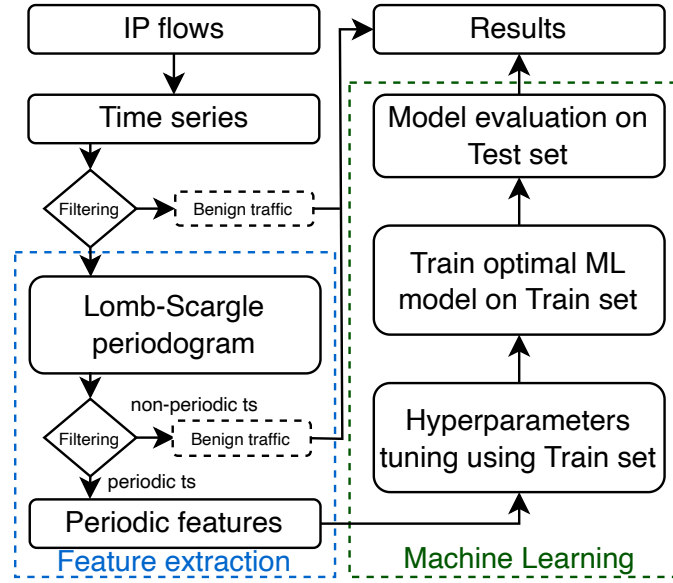


Figure 5.17: Methodology of botnet detection using periodicity behaviour in *Flow time series*.

To detect periodic behaviour in these time series, the Lomb–Scargle periodogram is employed along with SST test, using a *Significance level* of 0.1 and a *Percentage level* of 0.995. Only the time series deemed periodic by this test are retained for further processing.

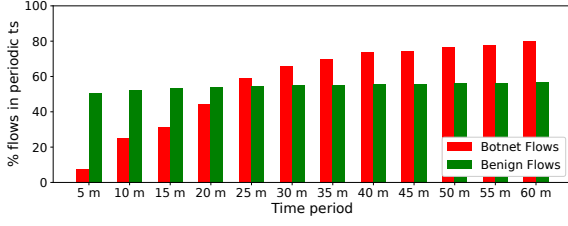
Next, feature engineering was performed on the periodic time series to generate a feature vector comprising 44 statistical metrics. These features provide a rich characterization of the observed network behaviour and include basic statistical, temporal, spectral, and periodic features.

These features collectively enable robust characterization of network dependencies. The CESNET-CC25 dataset already includes these extracted features, which were used directly as input to a machine learning pipeline. The pipeline outputs a binary decision indicating whether the periodic time series represents botnet C&C communication or benign activity.

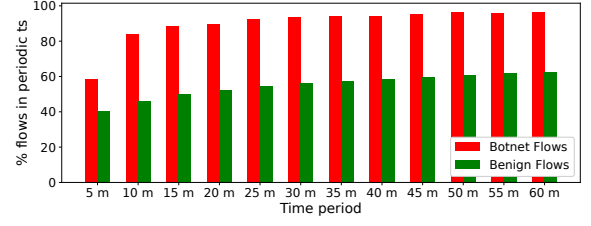
Crucially, there is an assumption that C&C traffic exhibits periodic patterns. Therefore, any time series with an insufficient number of data points or lacking significant periodicity is pre-classified as benign before the feature extraction stage.

Experimental evaluation

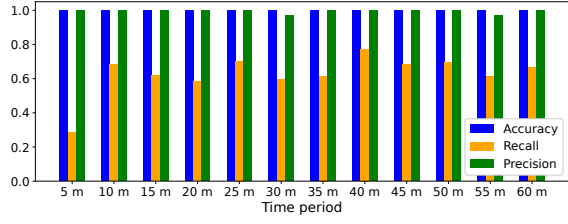
Multiple datasets of periodic features were constructed by varying the time window length $t \in [300, 3600]$ seconds. For each selected time period, periodic time series were extracted, and the performance of three machine learning algorithms commonly used for tabular data was evaluated: k-Nearest Neighbors (k-NN), Random Forest, and XGBoost. All models are evaluated on both datasets (CTU-13 and CESNET-CC25) across all time periods. Among the tested algorithms, XGBoost consistently achieved the highest performance on



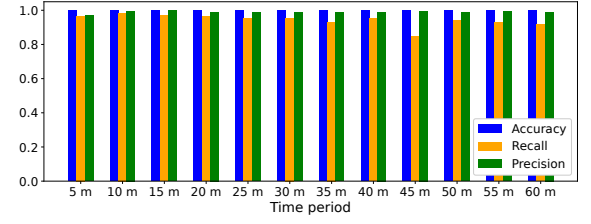
(a) Ratio of periodic IP flows in the CTU-13 dataset.



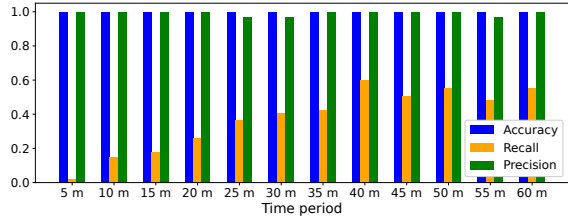
(b) Ratio of periodic IP flows in the CESNET-CC25 dataset.



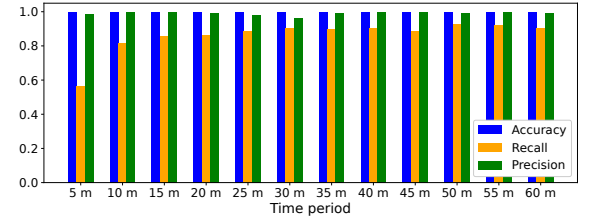
(c) Results of detection based on periodicity on the time series features for the CTU-13 dataset.



(d) Results of detection based on periodicity on the time series features for the CESNET-CC25 dataset.



(e) Results of detection based on periodicity transferred from the time series features to the IP flows for the CTU-13 dataset.



(f) Results of detection based on periodicity transferred from the time series features to the IP flows for the CESNET-CC25 dataset.

Figure 5.18: Experimental evaluation of detecting C&C communication through periodic behaviours

the validation set; therefore, the results reported in this paper are based on the performance of XGBoost on the test set.

The choice of the time window t significantly affects the results. As t increases, the number of detected periodic time series, as well as the proportion of IP flows contained within them, increases—this trend is shown in Figure 5.18a and Figure 5.18b. This effect is especially evident in the long-term CESNET-CC25 dataset, where almost 100% of C&C flows are captured within periodic time series as the time window grows. In contrast, the CTU-13 dataset includes multiple short-duration scenarios with a limited number of C&C samples, leading to many C&C flows being missed in the periodic time series extraction.

This discrepancy directly impacts the classification results presented in Figure 5.18c and Figure 5.18d. For both datasets, Accuracy and Precision exceed 99.9% across most

Table 5.5: Comparison of botnet detection methods on the CTU-13 dataset. Accuracy and F1-score are presented in %.

Related work	Used Features	Model	Accuracy	F1-score
Koumar et al. [A.3]	NetTiSA flow	XGBoost	99.95	99.79
Stergiopoulos et al. [226]	Flow-based	CART	99.85	99.90
Nugraha et al. [170]	Flow-based	CNN, LSTM	99.01	99.02
Jiang et al. [107]	Heuristic-based	Transfer learning	–	88.6
Chinta et al. [45]	Flow-based	Decision Tree	99.9	–
Gunawan et al. [78]	Flow-based	k-NN	97.2	–
Shamshirband et al. [210]	Statistical-based	HP-ELM	96.25	–
This approach	Flow Periodicity	XGBoost	99.99*	74.81*

* The results were obtained for the multiflow approach. Thus, the results were transferred to the flow by using the flow count in the multiflow time series as a weight.

time periods. However, Recall is significantly affected by the proportion of C&C flows in the periodic time series. On the CESNET-CC25 dataset, Recall consistently exceeds 90% for most time windows. Conversely, on CTU-13, Recall often remains below 60% due to the sparse representation of C&C communication in periodic time series.

The issue becomes more pronounced when the results are translated to the per-flow level. In this setup, all flows that are not part of a periodic time series are labeled as benign by default. This leads to a further drop in Recall on the CTU-13 dataset, as shown in Figure 5.18e and Figure 5.18f. Specifically, an average 25% decrease in Recall is observed for CTU-13, and even the best-performing configuration fails to reach 60% Recall. In contrast, CESNET-CC25 does not exhibit such a drop, confirming its robustness and suitability for periodicity-based detection.

This evaluation demonstrates that periodic C&C behaviour can be effectively detected in *Flow time series*. Importantly, when the model classifies a time series as C&C based on its periodic nature, the confidence is extremely high. The 100% Precision is consistently achieved across both datasets. This makes the periodic detection approach highly reliable for confirming the presence of C&C activity, even if it may under-detect sparse or short-lived attacks. Moreover, this approach addresses the issue that identical flows are shared between botnet and benign classes, as this phenomenon does not occur at the multiflow level.

Finally, the results are compared with existing work on the CTU-13 dataset in Table 5.5. While this method achieves comparable Accuracy, the F1-score is slightly lower, which can be attributed to the limited recall caused by sparse C&C presence in periodic flows, as discussed earlier. Nevertheless, this approach uniquely offers 100% Precision on CTU-13, underscoring its value for high-confidence botnet detection.

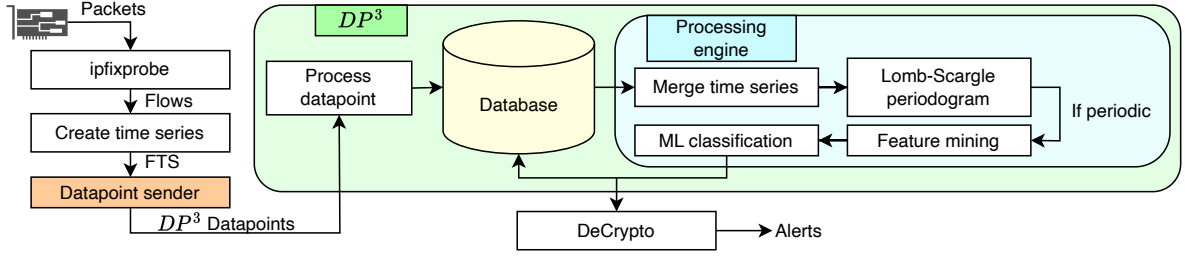


Figure 5.19: Deployment into high-speed ISP network using DP^3 platform

5.8 Deployment

The deployment of the periodic detection approach was proposed in the thesis author’s paper [A.5], where deployment is also addressed for the DeCrypto system. The DeCrypto is a system for detecting cryptomining at the flow level within the ISP network. In the thesis author’s paper [A.5] proposed the extension of the DeCrypto by long-term analysis of *Flow time series*. Therefore, the deployment of detection based on periodic behaviors was further addressed.

The *Dynamic Profile Processing Platform* (DP^3), available at GitHub⁸, is used for the deployment of the periodic detection approach into a high-speed ISP network. The DP^3 is an open-source data processing platform for maintaining dynamically changing profiles of *entities* (network dependencies in this case) represented by sets of attributes of different data types, including various types of time series. It enables the application of custom processing functions over profiles to enrich, correlate, or analyze the data in other ways, thereby deriving new information or detecting specific events.

The proposed deployment of the periodic detection approach with the DP^3 platform is shown in Figure 5.19. The *Create time series* module creates *Flow time series* and, using a *Datapoint sender*, it sends them into a DP^3 instance, where they are stored in profiles of corresponding network dependencies. The processing engine of DP^3 then runs modules for the detection of periodic behavior, feature mining, and ML-based classification. The alerts are in this example sent into the DeCrypto system by calling DP^3 API.

5.9 Key Findings and Contributions

This chapter presented research on network threat detection using *Flow time series* analysis to identify periodic behaviors and the development of optimized detection pipelines for high-speed infrastructures. The key findings in this chapter are:

1. Standard flow-based monitoring and existing packet-level extensions, such as SPLT, are often inadequate for long-term behavioral analysis because critical periodic patterns are lost during aggregation or truncated by flow timeouts.

⁸<https://github.com/CESNET/dp3>

2. Two distinct types of periodic behavior were identified: *Clear* behavior, characterized by consistent data volumes over stable time intervals (e.g., botnet heartbeats), and *Sinusoidal* behavior, where data volumes fluctuate despite periodic timing (e.g., social network background synchronization).
3. The Lomb-Scargle periodogram, combined with Scargle’s Significance Test (SST), provides a robust mathematical framework for detecting these periodicities in unevenly sampled *Flow time series*.
4. An enhanced detection pipeline was developed for high-speed networks, capable of processing 3 million flows in 5 seconds by utilizing histogram-based constant time series testing and optimized feature mining.
5. In a multiclass classification task for applications and operating systems, XGBoost achieved the highest performance with 91% accuracy and a 90% F1-score.
6. Periodic features were shown to be highly effective for cryptomining detection, with optimized models achieving up to 95.19% accuracy and a 92.21% F1-score across various time intervals.
7. The analysis of botnet Command-and-Control (C&C) communication revealed that while sparse or short-lived attacks may be under-detected, the periodicity-based approach achieves 100% precision across multiple datasets, making it a highly reliable tool for high-confidence threat detection.
8. The novel dataset CESNET-CC25 for detecting C&C communication in botnets was proposed.
9. Real-world feasibility was established through the integration of this methodology into the DP^3 processing platform, enabling deployment within high-speed ISP infrastructures for continuous entity profiling and alert generation.

The results in this chapter indicate a significant advancement in monitoring encrypted and automated network traffic. By applying TSA to multiflow records belonging to the same generating process, security tools can mitigate the loss of long-term behavioral patterns while maintaining the data volume advantages of flow-based monitoring. The developed methodologies provide a scalable bridge between traditional telemetry and the complex requirements of modern threat detection in high-speed environments.

Threat Detection using Aggregated Time Series Analysis

In this chapter, threats are detected using network traffic forecasting and anomaly detection techniques. For the evaluation of network traffic forecasting, real-world big data datasets are necessary. However, recent surveys have identified a lack of such datasets as a major challenge in this domain. Therefore, the recent thesis author's paper [A.7] published a dataset called CESNET TimeSeries24. This dataset was also used for address threat detection, utilizing traffic forecasting and anomaly detection. The dataset contains Aggregated time series in which several anomalies can be detected. The anomaly types are described in depth to get insight into which anomalies occur in real-world networks. Moreover, in the thesis author's paper [A.8], a system for detecting such anomalies in network traffic was proposed. However, detected anomalies can not only be a threat, but also an unusual activity which is not malicious at all, for example, an unexpected file download or user activity. It is crucial for Security Operation Centre (SoC) teams to evaluate if an anomaly is malicious or not. However, this is time and resource-consuming, and many organizations do not have a SoC team at all. Therefore, in the thesis author's paper [A.9], the method for autonomous evaluation of detected anomalies is proposed using Large Language Models (LLM).

6.1 Motivation

Network traffic monitoring plays a crucial role in network management and overall computer security [59]. Network-based intrusion detection and prevention systems can protect infrastructure against users' sloppiness, policy violations, or intentional internal attacks. The widespread adoption of traffic encryption significantly reduces visibility, making the maintenance of network security a challenging task.

As a result, gaining insights into encrypted network traffic has become essential, particularly for threat detection. Recent research has focused on detecting security threats through

the classification of encrypted traffic using machine learning techniques [A.3,4,11,143,181]. Supervised machine learning classification models can detect known attacks captured in the training dataset or those resembling them (e.g., malware from the same family). However, in the network traffic monitoring domain, there is a substantial challenge in acquiring up-to-date threat datasets [77]. Therefore, unsupervised anomaly detection plays a crucial role in network traffic monitoring [264] as it can detect unknown (zero-day) attacks based on behavioral changes caused by infection [17].

The unsupervised anomaly detection method assigns anomalous scores to network entity behavior based on patterns and characteristics learned from historical data. One of the most popular unsupervised anomaly detection algorithms is based on traffic forecasting, also referred to as network traffic prediction. An anomaly is detected by an algorithm when the difference between the forecasted value and observation exceeds a defined threshold. Traffic forecasting is useful for other networking use cases, such as data-driven network management, resource allocation, and service orchestration.

In recent years, there has been a rapid development in forecasting and anomaly detection methods, not limited to computer science. Wu et al. [262] attributed this development to the rise and successful use of neural networks. The recent performance improvement of forecasting methods for network traffic monitoring remains uncertain due to the absence of long-term datasets [66]. In their survey, Ferreira et al. [66] describe the lack of a reference dataset as the crucial obstacle related to performance evaluation. Additionally, real-world datasets used in the evaluation are not publicly available due to privacy concerns. Therefore, most publicly available datasets have synthetic origins.

Synthetic data does not necessarily represent real-world use. Wu et al. [262] show that novel anomaly detection and forecasting approaches evaluated on synthetic datasets lead to the illusion of nonexistent progress. The preferable option is to use up-to-date real-world data, like the MAWIlab project WIDE [46], which publishes anonymized packet captures daily. However, the WIDE project provides only brief 15-minute daily packet traces [67], which is an insufficient time window for effective traffic modeling.

Therefore, the thesis author's paper [A.7] proposes the CESNET-TimeSeries24 dataset to address these challenges by including long-term monitoring of selected statistical metrics for 40 weeks for each IP address on the Internet Service Provider (ISP) network CESNET3. Moreover, the thesis author's paper [A.8] proposed a deployable system for anomaly detection was proposed. Detection of anomalies (i.e., outliers, abnormalities) is crucial for maintaining network security. However, evaluating detected anomalies is not a trivial task for the Security Operation Centre (SoC) team due to the high number of anomalies that do not pose a threat (abnormal traffic but not malicious activity). Therefore, in the thesis author's papers [A.9], a novel approach is proposed that explains anomalies using Large Language Models (LLM).

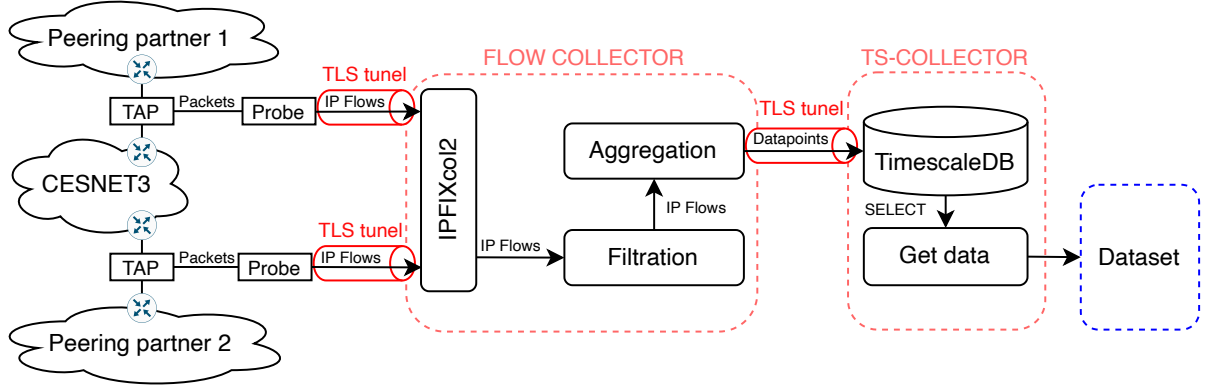


Figure 6.1: Architecture of dataset collection

6.2 CESNET-TimeSeries24 Dataset

In the thesis author’s paper [A.7], a novel CESNET-TimeSeries dataset was published. The dataset was obtained from the network traffic of the CESNET3 network—an ISP-like network providing internet access to public and research institutions in the Czech Republic. The network spans the entire country and serves approximately half a million users daily.

Data capture

Since ISP networks transfer huge volumes of data, packet-based monitoring systems are infeasible due to the costs of processing and storage. Therefore, the CESNET3 network is monitored using a standard IP flow monitoring system located at the perimeter. All transit lines to the peering partners are equipped with flow-monitoring probes.

The monitoring infrastructure for creating the dataset is detailed in Figure 6.1. Multiple Network Test Access Point (TAP) are before the edge routers of the CESNET3 network and mirror the traffic to monitoring probes, which are servers equipped with a high-speed monitoring card capable of handling 200 Gb/s. The flow exporter ipfixprobe is installed on the monitoring probe, and it collects the following features: start time, end time, number of packets, number of bytes, and Time To Live (TTL). None of the collected information contains application-level information to ensure the privacy of user communications is not compromised. The collected data are sent using IPFIX [48] protocol and secured (Transport Layer Security (TLS)) tunnel to the IP flow collector server with IPFIXcol2 (<https://github.com/CESNET/ipfixcol2>) flow collecting software installed.

Time series aggregation

The flow collector server contains filtration and aggregation modules as depicted in Figure 6.1. The filtration module removes all transient traffic where neither the source nor destination addresses belong to CESNET3, and the packets are merely passing through the network. All single TCP-SYN packet connections, i.e., scans, are also filtered out. Since

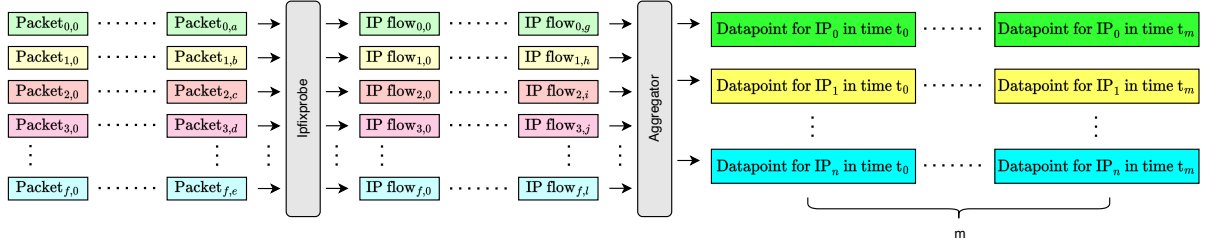


Figure 6.2: This diagram describes the aggregation process for capturing *Aggregated time series* from network traffic

the network probes are placed before the edge routers and firewalls, TCP scans make up the majority of the collected flows. Too many scans would cause significant noise in the time series and result in possible anomaly detection false negatives. Since simple methods [32, 225] can detect the scans, scans are excluded from the dataset and filtered out. The filtration module passes all other flows to the aggregation module.

Evenly spaced time series are created for each IP address by aggregating IP flow records into datapoints based on a time window. The process diagram for network traffic processing and aggregation is shown in Figure 6.2. The created datapoints represent the behavior of IP addresses within a defined 10-minutes time window. The vector of time-series metrics $v_{ip,i}$ describes the IP address ip in the i -th time window. Thus, IP flows for vector $v_{ip,i}$ are captured in time windows starting at t_i and ending at t_{i+1} . The aggregator module sends aggregated datapoints using a secured (TLS) tunnel into the Time Series collector. The datapoints are stored in TimeScaleDB¹. Time series are built by selecting datapoints from the database.

Datapoints created by the aggregation of IP flows contain the following time-series metrics:

Simple volumetric metrics: the number of IP flows, packets, and bytes

Unique volumetric metrics: the number of unique destination IP addresses, destination Autonomous System Numbers (ASNs), and destination transport layer ports

Ratios metrics: the ratio of UDP/TCP packets, UDP/TCP transmitted data size, packets direction, and bytes direction

Average metrics: the average flow duration, and Time To Live (TTL)

The aggregation of *Unique volumetric metrics* is memory-intensive since all unique values must be stored in an array. A server with 41 GB of RAM was used. The server was enough for 10-minute aggregation on the ISP network.

¹<https://www.timescale.com>

Anonymization

The capturing process started on 9. October 2023. On 14. July 2024, after 40 weeks, data capturing ended, and the time series was extracted from the database. As part of the database framework, a script automatically adds a corresponding institution and an institution subnet for each IP address. The IP addresses, institution names, or exact institution subnets were not stored. Instead, randomly assigned numbers were used as identifiers for IP addresses, institutions, and institution subnets during database creation, which effectively anonymizes the data. Without access to the mapping table within the database framework, this step cannot be reversed to reconnect the time series with a particular IP address, institution, or institutional network.

Data preprocessing

Preprocessing steps included: *i)* time series filtering, *ii)* creation of multiple time aggregation, *iii)* creation of time series for institutions, *iv)* creation of time series of institutional subnets, and *v)* adding information about weekends and holidays.

Time series filtering: The raw dataset obtained from the database has time series data for approximately 400,000 IP addresses. However, many of these network entities showed very low activity—most of the time, they were inactive, leading to largely empty records in the 10-minute aggregation. To further reduce the dataset size, unusable time series were removed by filtering out those with more than 99.75% of their time intervals empty (i.e., containing 100 or fewer non-empty data points). In comparison, studies focused on forecasting with missing values often consider 50% missing data as an extreme case [237, 263]. Applied approach, therefore, eliminates only time series that are effectively unusable for forecasting. This filtering step reduced the dataset size to 275,124 IP addresses.

Multiple time aggregation: Datapoints aggregate network traffic into 10-minute intervals. The size of the aggregation interval influences anomaly detection procedures, mainly the training speed of the detection model. Recognizing 10-minute intervals can be too short for longitudinal anomaly detection methods. Therefore, one-hour and one-day aggregation intervals are provided. These intervals were aggregated from the 10-minute interval using the sum of values. In certain cases, such as with metrics that contain a number of unique values, a simple summation would result in information loss. To address this, three additional time-series metrics were provided for each original one: sum, mean, and standard deviation. Finally, the mean is used for the remaining time series metrics.

Time series of institutions: The 283 unique institution IDs are exported from the database to group the IP addresses based on institution. These can provide the overall network traffic perspective required for the identification of many security events.

Time series of institutional subnets: Many institutions have multiple networks, or subnets, for different locations or for different purposes in the organization. Network administrators often handle security for the overall network and for each subnet. To accommodate that, the IP addresses were divided into groups based on institution subnets in the same location using information from the CESNET association. The 548 institution subnets were identified inside the CESNET3 network.

Weekend and holidays: Network traffic is different on weekends and holidays, making it crucial for forecasting and anomaly detection model training and evaluation. To support this, the dataset includes weekend dates and dates of public holidays in the Czech Republic.

6.3 Anomalies in Network Traffic

A large number of anomalies are usually present in the network traffic. These anomalies can be detected using standard or novel methods. The anomalies were evaluated in the CESNET-TimeSeries24 dataset to determine which anomaly types are present in the network traffic, and, if applicable, to assess whether the time series metrics are suitable for anomaly detection.

Network anomaly is an overused term in many works and propagation materials, and can be understood as an incident that marks malicious traffic. This terminology is in this domain commonly used in malware traffic detection. This thesis interprets the anomaly differently and employs a stricter designation: an outlier. An intuitive definition of an outlier: a data point that deviates significantly from other observed data points, enough to raise suspicion that it was generated by a different mechanism [85]. The formal definition of an anomaly as used in this thesis is:

Definition 6.3.1 (Anomaly). A *Anomaly* (i.e., Outlier, Abnormality) is an observation $x_i \in \mathbf{X}$ for which it is true:

$$P(x_i) < \epsilon \tag{6.1}$$

where $\mathbf{X}$ is a set of observations, $P(x_i)$ is the density estimation of observing x_i and ϵ is a threshold.

The point is that the estimation function $P(x)$ and parameter ϵ are unknown; they heavily depend on the data (i.e., the set of observations $\mathbf{X}$) and the use case.

Anomaly Types in Network Traffic

There are many types of anomalies described in the literature. Many of these anomalies are present in this dataset. Examples of the anomaly types frequently occurring in the dataset are shown in Figure 6.3.

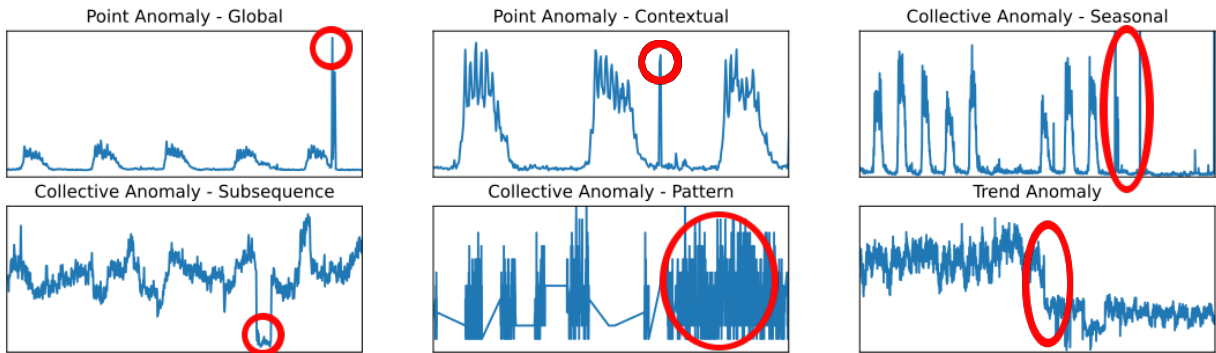


Figure 6.3: Examples of the anomaly types occurring in the *Aggregated time series* from network traffic

The first type of anomaly is a *Point Anomaly* [40], a single data point that significantly deviates from the rest of the datapoints in the time series. There are two types of point anomalies:

Definition 6.3.2 (Global anomaly). A data point that deviates significantly from the overall pattern or distribution of the entire dataset is called an *Global anomaly*. It is an extreme value when compared to the rest of the data.

Definition 6.3.3 (Contextual anomaly). A data point that is an outlier within a specific context or condition but not necessarily when viewed in a different context is called a *Contextual anomaly*.

The second type of anomaly is a *Collective Anomaly* [40], a sequence of datapoints that is anomalous when considered together, even if individual points might not be. There are two types of collective anomalies:

Definition 6.3.4 (Seasonal anomaly). A contiguous subsequence of datapoints that is anomalous compared to the seasonal pattern of the time series is called a *Seasonal anomaly*.

Definition 6.3.5 (Subsequence anomaly). A contiguous subsequence of datapoints that is anomalous compared to the rest of the time series is called a *Subsequence anomaly*.

Definition 6.3.6 (Pattern anomaly). A sequence of datapoints that together form an unusual pattern that does not conform to known time series patterns is called a *Pattern anomaly*.

The third type of anomaly is a *Trend Anomaly* [26], an unexpected change in the trend of the time series data, such as a sudden shift from a positive to a negative trend. Similar data behaviors in data science are referred to as data or content drifts [277], and sometimes, this anomaly type is called a *Drift Anomaly*.

Security Incident Analysis using Anomalies

When analyzing a detected anomaly from the network traffic, it is crucial to explore the metrics that characterize the anomaly. For example, CESNET experts scrutinized metrics in detail for the anomaly detected on time series from the CESNET-TimeSeries24 dataset with IP address ID 1367, as illustrated in Figure 6.4, to identify the event as a Denial of Service (DoS) attack.

The analysis reveals several key observations. Both the flow and packet counts (metrics *n_flows* and *n_packets*) increase significantly, yet they rise in parallel, indicating a one-packet flow characteristic. Despite the increase in flows and packets, the transmitted data (metric *n_bytes*) does not rise correspondingly, suggesting that the packets involved are small in size. Furthermore, the number of destination ports (metric *n_dest_ports*) remains steady, dismissing the possibility of a scan. A significant shift in the TCP/UDP ratio (metric *tcp_udp_ratio_bytes*) is observed, with almost all the anomalous traffic being UDP, further supporting the DoS identification.

Additional metrics reinforce this conclusion. The traffic direction (metric *dir_ratio_bytes*) is near zero, indicating nearly all the anomalous traffic is directed toward the monitored IP address. The average duration of the flows (metric *avg_duration*) is close to zero, consistent with one-packet flows typically seen in DoS attacks. The Time To Live (TTL) values (metric *avg_ttl*) remain nearly constant during the anomaly, implying that the traffic likely originates from a single source. Finally, the number of destination IPs (metric *n_dest_ips*) shows a slight decrease, reinforcing the idea of a singular sender and confirming the effectiveness of the DoS attack. These combined metrics clearly point to a DoS event targeting the monitored IP address.

6.4 Network Traffic Forecasting using SARIMA

The forecasting-based approaches are crucial for threat detection because they can be employed for anomaly detection. A proper forecasting model can forecast future values, which can be compared with real observations using confidence intervals to decide if the observation is an anomaly (i.e., unexpected by the model).

For demonstration, the time series from the CESNET-TimeSeries24 dataset with IP address ID 103, the number of IP flows, and a one-hour aggregation interval were selected. The basic Seasonal Autoregressive Integrated Moving Average (SARIMA) is applied to time series as a forecasting model with the order equal to (1, 1, 1) and seasonal order equal to (1, 1, 1, 168). This means that each of the three components of the model has the same weight in the modeling, and the seasonality was set to one week (168 datapoints). The SARIMA was trained on the monthly data (744 datapoints). In this example, the two prediction intervals, 7 days and 2 days, are used to demonstrate the difference. The model was continually retrained after each prediction.

The result of this forecasting demonstration is shown in Figure 6.5. The Figure contains 7-day and 2-day prediction intervals, and the model retraining after 2 days improved the

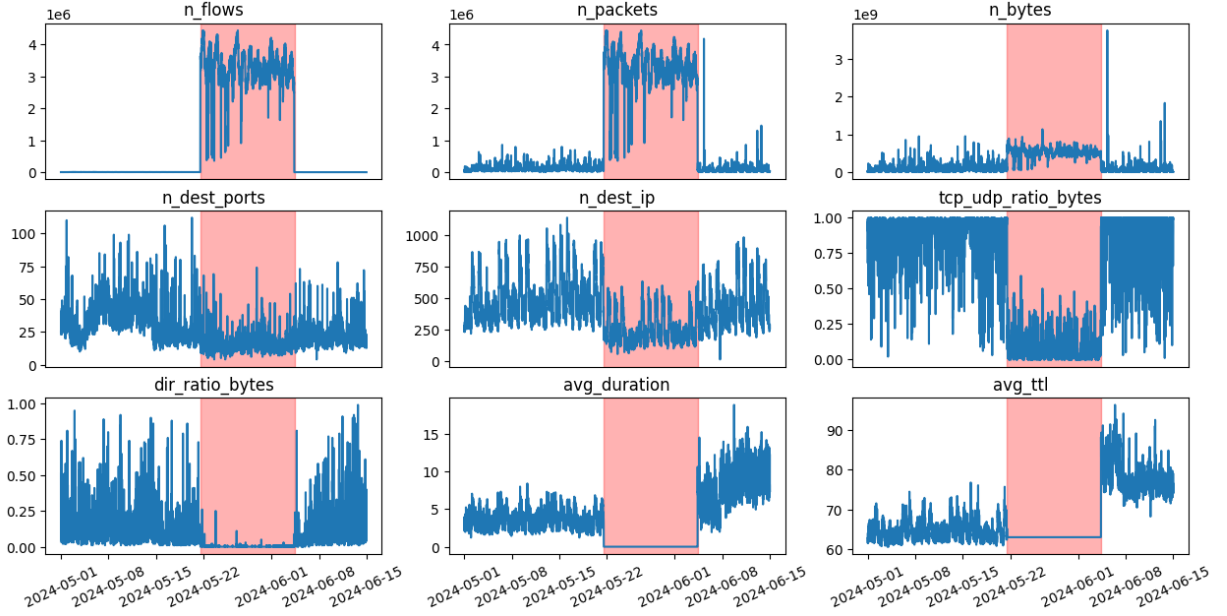


Figure 6.4: Analysis of detected anomaly on time series IP address ID 1367 identified as DoS by CESNET experts

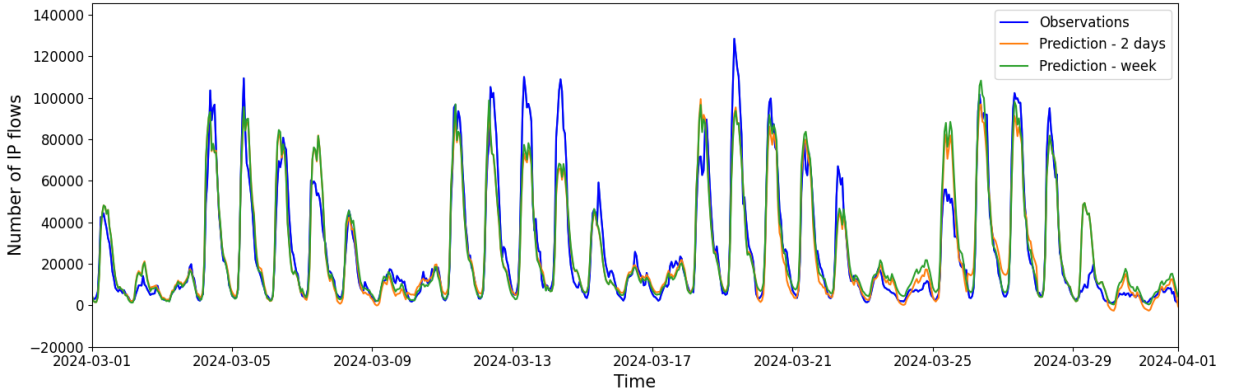


Figure 6.5: Example of using SARIMA model with order $(1, 1, 1)$ and with seasonal order $(1, 1, 1, 168)$ for forecasting of number of IP flows for the time series of IP address ID 103.

results. The results are compared by using the evaluation metrics Root Mean Square Error (RMSE), Symmetric Mean Absolute Percentage Error (SMAPE), and Coefficient of determination (R^2 Score). For all evaluation metrics used, a lower value represents better forecasting performance. The predicted 7-day data achieves an RMSE of 10,951.26, a SMAPE of 40.66, and an R^2 Score of 0.77. Whereas the predicted 2 days of data achieves 10,293.81 RMSE, 40.86 SMAPE, and 0.79 R^2 Score. In comparison to the predicted 7-day forecast, the predicted 2-day forecast RMSE and R^2 Score were better, but the SMAPE was slightly worse.

6.5 Anomaly Detection System

Anomaly detection in network traffic is crucial for identifying abnormal patterns that may indicate security breaches, faulty systems, or unexpected behaviors. Methods based on forecasting often involve predicting future network behavior using historical data and flagging any significant deviations from these predictions as outliers. Techniques like Autoregressive Integrated Moving Average (ARIMA) and Long short-term memory (LSTM) models are commonly used for this purpose, as they can capture time dependencies in the data. By comparing actual traffic to forecasted values, these methods can effectively detect unusual spikes or drops in network activity, signaling potential outliers or threats.

Although these anomaly detection methods exist and have been widely examined in related works, deploying them is a crucial and challenging task. Therefore, in the thesis author's paper [A.8], a novel modular Network Outlier Detection System (NODS) was proposed. The NODS is built from open-source software and enables the deployment of outlier detection methods based on the forecasting of network traffic in real-world scenarios. The NODS was tested on the real-world ISP network CESNET3, where the system was in the testing phase for several months to ensure stability and reliability of the system.

At the beginning of the network outlier detection system, a set of requirements that such a system should fulfill was defined based on previous experience and experiments in the area. The requirements are the following:

1. Architecture design should be modular, supporting model management, training, and predicting the feature values. Moreover, the system should provide an interface for running multiple instances of the same or different models with varying hyperparameter settings.
2. Multiple aggregation intervals should be supported since more than one aggregating interval run in parallel might reveal different anomalies.
3. The system should support different aggregation (time-series) metrics.
4. Several types of aggregation identifiers should be supported, e.g., IP addresses (devices), network prefixes, or IP-to-IP dependence.
5. Resource requirements should be kept minimal as the system is intended to operate in large ISP networks.

System Architecture

The NODS is composed solely from open-source software listed in Table 6.1. IP flow exporter Ipfixprobe is used to aggregate network packets into flows and send them to the flow collector IPFIXcol2. NODS is deployed together with the flow collector, and both of them utilize the NEMEA framework [38].

The NODS is a set of NEMEA modules written in Python that communicate via the NEMEA interface. The design of the system is depicted in Figure 6.6. The *Aggregator*

Table 6.1: Open source software used in the NODS system.

Name	Version	GitHub
ipfixprobe	4.11.0	CESNET/ipfixprobe
IPFIXcol2	2.2.1	CESNET/ipfixcol2
NEMEA Framework	0.14.0	CESNET/Nemea-Framework
NEMEA modules	2.20.0	CESNET/Nemea-Modules
NEMEA Supervisor	1.8.2	CESNET/Nemea-Supervisor
TimeScaleDB-14	2.15.0	timescale/timescaledb

module receives flows on the input NEMEA interface and aggregates them based on the set aggregation interval and identifier. The IP flows are not stored in the memory. The time series metrics are computed by using intermediate computation.

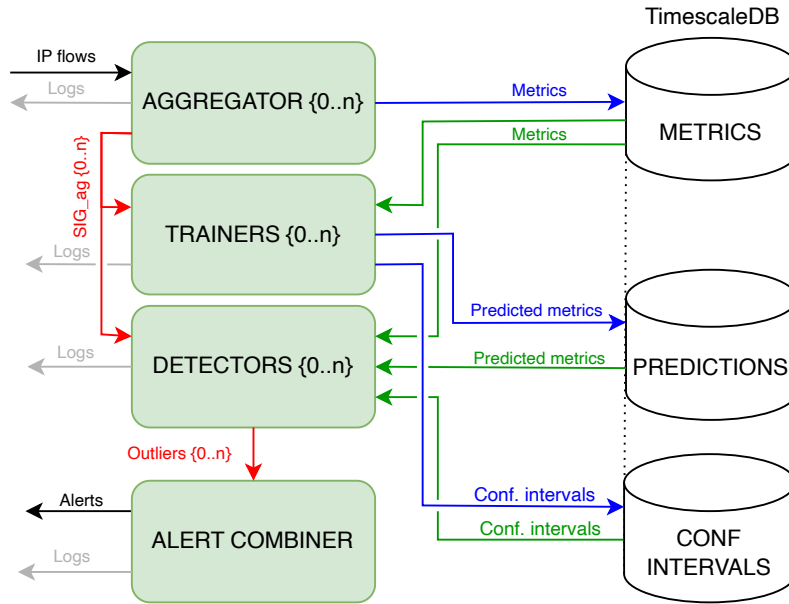


Figure 6.6: Design of the NODS architecture.

The datapoint is a set of time series metrics captured between the start and end of the aggregation interval for the specific identifier. After each aggregation interval, the *Aggregator* creates a new thread that inserts created data points into the TimescaleDB database. The main thread of the *Aggregator* continues receiving flows and creating new data points. Moreover, after the new datapoints are inserted into the database, the child thread of the *Aggregator* sends an echo with a timestamp of the datapoints to the output NEMEA interface.

The echo from the *Aggregator* is received by two NEMEA modules *Trainers* and *Detectors* using their input NEMEA interfaces. The *Trainers* load all enabled outlier detection

models and their configuration. Then, the training phase for each model occurs. The model training function usually selects datapoints for each identifier from the database and performs model training. Trained models forecast future datapoints with confidence intervals stored in the database. The training process is usually time-consuming. Therefore, it is necessary to choose the appropriate training phase time.

The *Detectors* receive the echo and run each model detection function. The detection function selects observed datapoint x_i , predicted datapoint p_i , and confidence interval c_i for each identifier. The selected values are compared, and if $x_i < p_i - c_i$ or $x_i > p_i + c_i$, then the x_i is marked as an outlier. All detected outliers are reported via the output NEMEA interface.

The system supports multiple aggregation interval settings by running an instance of the previously described combination of modules for each interval (e.g., one for 10 minutes and another for 1 hour).

The *Alert Combiner* receives outliers from all aggregation intervals and all models and combines them into the final output for the SoC team. The outliers are combined per each identifier into incidents. Moreover, incidents are divided into two categories—critical and others. The critical incidents are sent as alerts to the SoC team, while others are stored for potential future analysis.

Anomaly Detection Models

The outlier detection models used in the NODS system are based on statistical analysis of time series from network traffic. These models depend on two fundamental windows: training and prediction windows. The training window collects historical data over a specific period, such as one month, week, etc., which the model uses to learn patterns. The prediction window then forecasts future behavior based on this learned history, and the predicted values are compared to actual traffic observations. A threshold is applied to determine whether a statistical anomaly has occurred.

Moreover, the models must be time and memory sensitive to be deployable into high-speed ISP networks. Therefore, the NODS contain models that prioritize deployability over precision.

Mean Outlier Detection

The first of the selected outlier detectors is a model based on the average value of observed variables; Mean Outlier Detector (MOD). The model is based on the assumption of the normal distribution; thus, 99.9% of values from a set of values are located within the interval $(\mu - 5\sigma, \mu + 5\sigma)$, where μ is the mean of the values and σ is the standard deviation of the values. Therefore, MOD assumes the absolute validity of this assumption and checks whether the observed value complies $|x - \mu| > 5\sigma$.

Differential Outlier Detector

The Differential Outlier Detector (DOD) monitors the differences between the next values of the observed variable. It assumes that intermediate values should not change rapidly unless the change is part of a recurring pattern. If sudden shifts in values are detected, DOD identifies them as statistical anomalies. This approach makes it effective in detecting sharp changes that may be unseen by other models focusing on absolute values.

One limitation of DOD is that it can trigger frequent alerts, as the system does not store past outlier information. This means that when an outlier unexpectedly returns to normal, it may still be flagged. Although these alerts are not considered false positives, they can be considered redundant. A key difference between DOD and MOD is that DOD can catch quick changes, even if the values remain within the normal range, whereas MOD only detects anomalies if values exceed the threshold. Thus, DOD and MOD work together within the system.

SARIMA

The ARIMA model is a widely used method for anomaly detection, combining Autoregressive (AR) and Moving Average (MA) components to handle time series. It is useful for detecting patterns in time series data without seasonal trends [7]. The SARIMA model extends ARIMA by accounting for seasonality, making it particularly effective for network traffic, which often exhibits periodic patterns such as lower traffic on weekends [126].

Although SARIMA requires more parameters and can be challenging to configure, it is a more accurate model for detecting periodic or seasonal phenomena in network traffic. The major disadvantage of SARIMA is the time-intensive training process, which limits its deployment across multiple monitored IP addresses. In the NODS, SARIMA is configured with parameters that account for short-term seasonal behaviors.

Deployment Evaluation

This section provides a demonstration and evaluation of the system deployment in a real network environment. The NODS was deployed on the CESNET3 network, which connects major institutions nationwide through optical links, offering transmission speeds ranging from 1 to 400 Gbps.

The system was set to detect outliers in two parallel aggregation intervals—10 minutes and 1 hour—utilizing five time-series metrics—the number of flows, packets, bytes, unique IP addresses, and unique transport layer ports. The system was running in a virtual environment with 8 CPU, 16 GB RAM memory, and 100 GB drive. The demonstration and evaluation period presented in this work was one month and focused on anomaly detection for security purposes in a SoC.

Detected Outliers

The overall number of outliers detected during the evaluation period is analyzed to gain an understanding of how the NODS works in real-world deployments. This evolution of outliers detected per each detection module for all monitored IP addresses and two aggregation intervals is depicted in Figure 6.7.

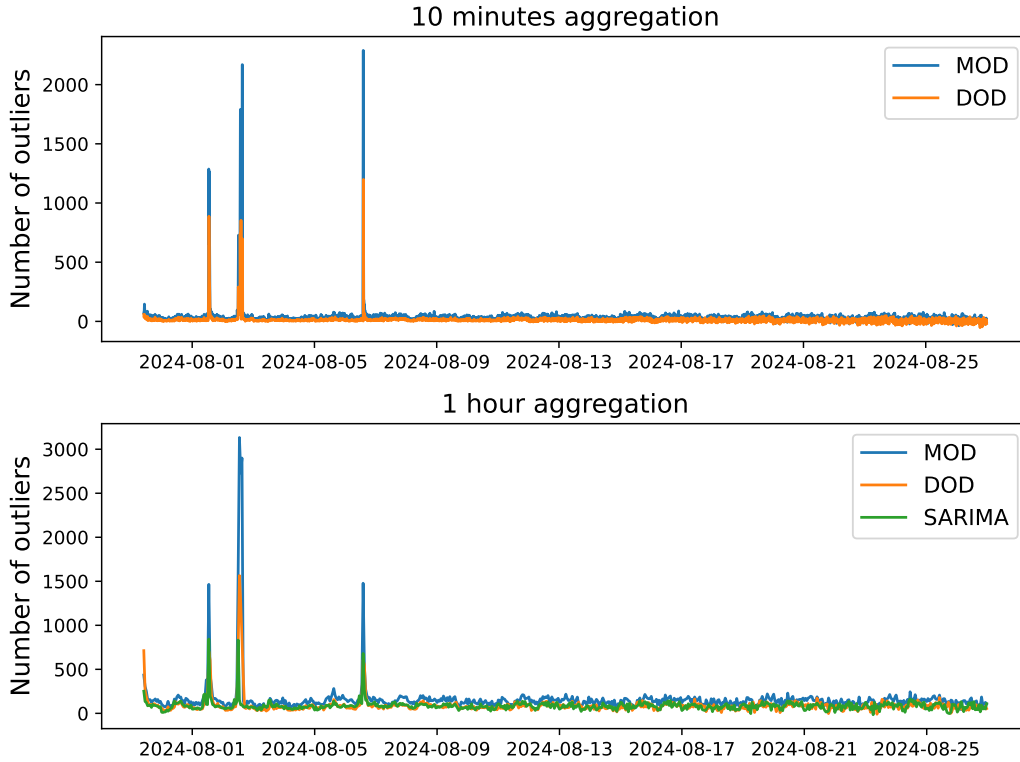


Figure 6.7: The evolution of detected outliers for each model.

The outliers of each model and aggregation interval are analyzed independently by the system. Over the evaluation period of one month, the models produced, on average, 375 outliers hourly, while MOD was responsible for 185, DOD for 100, and SARIMA for 90. However, three transients can be noticed in the graph where all models in all intervals logged significantly more outliers than usual. After the investigation, it was found that each of the transients is caused by the increase in the number of outliers for IP addresses from one specific network prefix.

The distribution of detected outliers is evaluated as presented in Figure 6.8. The Cumulative Distribution Function (CDF) curve shows that only approximately 10 to 15 percent of IP addresses are responsible for almost all detected outliers. A deeper analysis of detected IP addresses revealed that there are three groups of devices with the highest number of outliers. The first group is represented by user devices connected to WiFi endpoints. The second group consists of devices like routers, firewalls, and other core

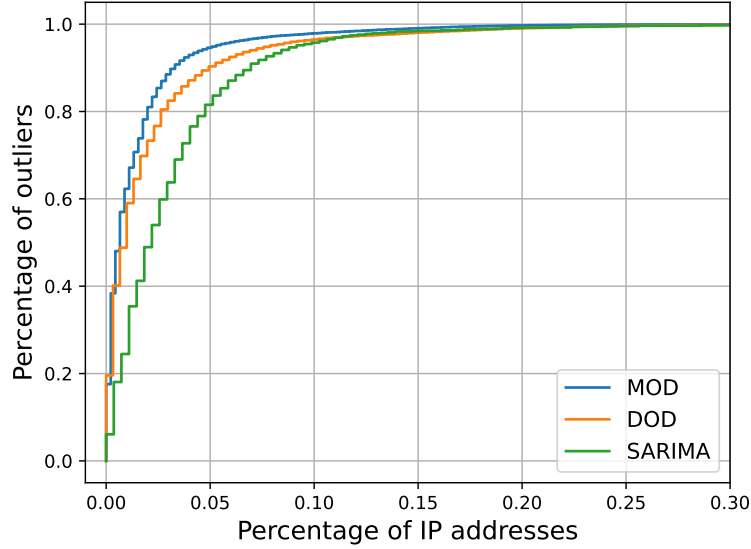


Figure 6.8: The relationship between the percentage of IP addresses and the percentage of detected outliers.

devices, which communicate sporadically. Thus, the time series is full of zero values, which causes false positives in detection when they start to communicate. The last group is created by disk and repository servers as their time series network traffic data contains unexpected file transfers.

More detailed information on the detected outliers is provided in Table 6.2. It can be seen that most outliers were detected in the number of flows per aggregation interval, and the least in the number of unique destination IP addresses. Moreover, Table 6.2 also shows the ratio between outlier and normal data points. The NODS system usually achieves a very small percentage of detected outliers, which is sufficient for the mathematical definition of an outlier.

It can be seen that the system provides a high number of detected outliers. However, the SoC team can not investigate every outlier. Moreover, if the number of outliers to investigate is large, the SoC team would rather ignore them or filter them out. Therefore, it is required to provide a ranking for the outliers and filter only the most critical ones that deserve further investigation.

In reaction to this, the outliers are grouped into so-called incidents. The incident is one or more outliers detected simultaneously over different aggregation intervals or metrics. Additionally, critical incidents were defined as incidents for which all models mark a timestamp of at least one 10-minute interval in a particular hour, and all models mark the corresponding hour interval as an outlier. Nevertheless, all models must consider the timestamp to be an outlier with 100% confidence. The critical incidents are then sent as alerts to the SoC team, while the rest of the incidents are logged for optional future analy-

Table 6.2: The number of outliers per time series metric in deployment for 100,000 IP addresses for one month (31 days). Moreover, in this deployment, NODS deploys the time series metrics—the number of flows, packets, bytes, unique IPs, and unique ports.

Aggregation interval	Time series metric	# outliers	% datapoints
10 min	n_flows	281,232	0.063
	n_packets	89,280	0.020
	n_bytes	93,744	0.021
	n_dest_ip_pub	17,893	0.004
	n_dest_ports	17,568	0.004
1 hour	n_flows	151,776	0.204
	n_packets	104,904	0.141
	n_bytes	98,952	0.133
	n_dest_ip_pub	12,648	0.017
	n_dest_ports	20,088	0.027

sis. The number of incidents and critical incidents is shown in Figure 6.9. The distributions show that for most IP addresses, a few incidents were detected. However, there is also an IP address for which more than 150 incidents were detected. This IP address is assigned to devices on WiFi; therefore, models cannot do proper prediction because IP addresses change for devices over time. Moreover, in Figure 6.9, it is shown that the number of crucial incidents is significantly smaller compared to all detected incidents. Furthermore, the number of crucial incidents is small enough for the average SoC team to explore.

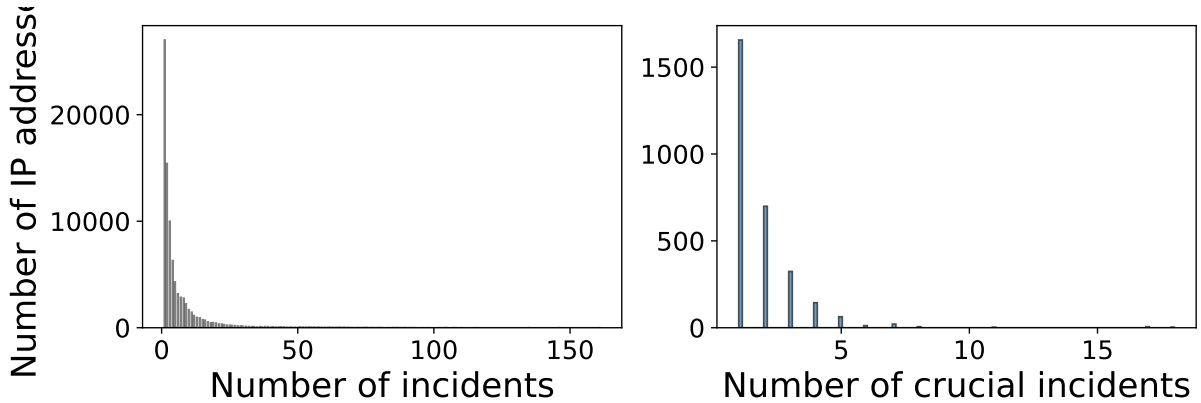


Figure 6.9: Comparison of the number of incidents and critical incidents.

6.6 Explainable Anomaly Detection

Network monitoring is a crucial part of any organization’s security strategy. Small organizations often cannot afford dedicated teams that observe network data and spot anomalies or security incidents. Larger organizations may have specialized teams, but many reported

incidents often overload these teams, causing *alert fatigue* [122]. They need to evaluate each alert, determine which to prioritize, and use extra data sources to classify or rank them. The described process might be inconsistent, especially if less experienced staff do not receive clear explanations.

Organizations like ISP face even more challenges. They often have limited information about traffic coming from other networks, which complicates the reasoning behind why anomalies occur. Concurrently, the amount of encrypted network traffic is increasing, and the volume of data passing through networks continues to grow daily. Consequently, numerous teams rely on time series volumetric data, which quantifies the traffic flow over time. An anomaly detection system notifies operators when it finds unusual spikes or drops. Unfortunately, these systems typically provide only superficial alerts, rather than detailed explanations. Even seasoned analysts may struggle to understand the true cause of an anomaly by simply examining the data, and too many false positives can lead some organizations to disable these systems altogether.

Recent research has shown that LLM offer promising ways of enhancing the interpretability and estimating the root causes of the anomalies [42, 252]. By identifying potential root causes, LLMs can assist both junior and senior analysts in determining why certain traffic patterns occur, thereby helping to classify the incident type, assess its importance, and inform more efficient responses. Moreover, the LLM can serve as an independent analyst agent, filtering alerts automatically based on the reasoning result.

Therefore, the thesis author’s paper [A.9] proposes an approach for using LLMs to explain already detected anomalies in network traffic time series volumetric data. Instead of relying on LLMs to identify anomalies in time series data, as for example in [140], an existing Machine Learning (ML)-based anomaly detection framework is used to flag suspicious events, and then the LLM explains these alerts. This setup reduces the need for heavy computation because the LLM is only used when an anomaly has already been identified. In the thesis author’s paper [A.9], a zero-shot method is used, meaning the LLM relies on its internal, general knowledge without extra training on specific data.

To demonstrate how this works, three different scenarios are presented using time series from the CESNET-TimeSeries24 dataset. Network alerts originate from a real anomaly detection system and are ideal for demonstrating how the use of LLM can provide more informed decisions and deeper insights. Overall, this approach makes anomaly detection more transparent and actionable for network operators, thereby reducing false positives and leading to stronger and more efficient cybersecurity practices.

Evaluation Methodology

The focus is on enhancing the explainability of anomaly detection in volumetric network traffic time series using LLMs. The methodology follows the process illustrated in Figure 6.10, which consists of multiple stages:

- 1. Volumetric Network Time Series** – Raw network time series data is obtained from the CESNET-TimeSeries24 dataset [A.7], which already includes preprocessed data

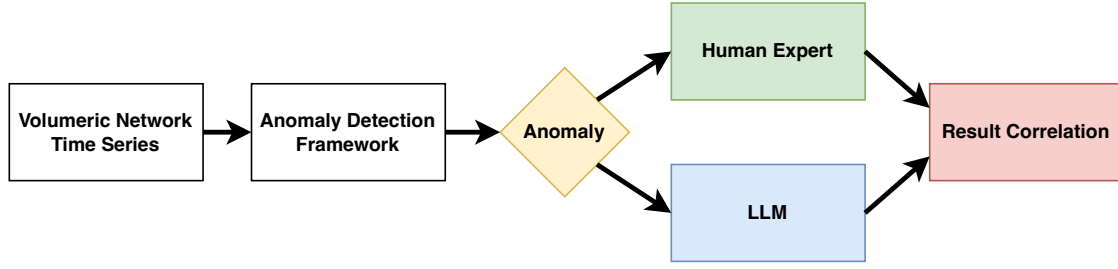


Figure 6.10: Holistic overview of evaluation methodology of explainable anomaly detection using LLM.

from a real network.

2. **Anomaly Detection** – Anomalies are identified using the State of the Art (SOTA) anomaly detection methods.
3. **Expert Annotation** – Three human experts in the network operation domain analyze and agree on the label of detected anomalies to establish ground truth.
4. **LLM-based Explainability** – The labeled anomalies are fed into an LLM, which generates explanations and insights.
4. **Evaluation & Correlation** – The LLM-generated outputs are compared with expert assessments to evaluate the model’s effectiveness in reasoning about network anomalies.

This structured approach ensures a systematic analysis of the explainability potential of LLMs when applied to network security tasks. To evaluate the effectiveness of LLMs in explaining network anomalies, the focus is taken on three distinct cases, each addressing different aspects of dealing with anomalies. While the core methodology remains consistent across these cases, the key differences arise from variations in the design of prompts and the selection of data. By exploring multiple scenarios, the adaptability and reliability of LLMs were assessed in helping network operators understand network disturbances.

Volumetric Network Time Series Data

The CESNET-TimeSeries24 dataset consists of a rich set of traffic features aggregated into time series at three different time granularities: 10 minutes, 1 hour, and 1 day. This work primarily utilizes a 1-hour aggregation interval, as it provides a human-interpretable representation of network activity while capturing behavioral trends over the entire 40-week period.

Although the dataset contains numerous real-world incidents, anomalies, and behavioral changes—including certain cyberattacks—it does not yet include predefined labels for anomalies. The labeling process remains an area for future research. However, this limitation does not prevent the dataset from being used for anomaly detection and explainability.

Given the extensive nature of the dataset, a subset of 70 detected anomalies of different kinds was selected for demonstration purposes, from which 53 were held out for evaluation. To enhance model interpretability, additional derived metrics were included, such as transformations of original metrics, ensuring a more contextual understanding of anomalous behavior.

Anomaly Annotation

Since the dataset lacks predefined labels, a human-in-the-loop annotation approach was adopted. Three domain experts specializing in network traffic monitoring manually examined the detected anomalies to determine their nature. Each outlier was classified as either a false positive or a legitimate anomaly (e.g., an attack, behavioral shift, or outage).

This manual annotation process closely simulates real-world network operation workflows, where human analysts investigate network traffic anomalies to decide whether further action is necessary. By leveraging their expertise, the annotators considered multiple traffic metrics to confidently infer the potential causes of incidents. For evaluation, 53 anomalies were held out, with 30 labeled for further inspection and 23 left as false positives and unimportant by the experts.

Prompt Tuning

To efficiently guide the LLM in explaining detected anomalies, a prompt tuning phase was conducted using an initial sample of 10 annotated anomalies. The remaining labeled anomalies were reserved for final evaluation. The prompt tuning phase involved searching for the optimal way to instruct the deployed general LLM model, using the detected anomaly context, to respond in a specific format. Instruction tuning was aimed at minimizing token usage while ensuring clear and actionable explanations.

Unlike Liu et al. [140], who inserted time series directly into LLMs for anomaly detection, anomaly timestamps were already predefined. Consequently, prompts focused on providing contextual information rather than anomaly identification. Each prompt included:

- Instruction of how the model should behave, how the data was constructed, what their general meaning is, and how the anomaly detection method worked.
- A value in the detected anomaly data point with a mean and standard deviation of 48 hours and 7 days time window before and after the data point, and information whether an anomaly was detected for the metric. Such data was provided for each metric.
- Time specification of the endpoint (time during the day, date, day of week, and if a public holiday).

- Expected structure of the prompt response (request to provide: priority of the event, model confidence of the response, short label, description, and instruction on the action that should be taken).

The prompt was intentionally created without explicitly providing knowledge of any thresholds, an explanation of what an anomaly means, or an understanding of what an anomaly can mean or how serious it can be. The models were evaluated for their zero-shot internal capability to address this problem.

The effectiveness of different LLMs in explaining anomalies was evaluated against expert-labeled data. Prompt tuning was conducted using GPT-4o, and the optimized prompt was subsequently tested across multiple LLMs to assess robustness and generalizability. The actual prompts and other materials related to this article are publicly available on GitHub².

Evaluation

LLM-generated responses were evaluated using OpenAI and Google AI Studio interfaces. The selected models included GPT-4, GPT-4o, GPT-4o-mini, and Gemini 2.0 Flash Experimental. All prompts were applied in separate, stateless conversations to ensure unbiased responses, preventing the models from retaining any prior context. The results are based on the 53 anomalies that were held out for evaluation, with each model questioned only once with all 53 samples.

The accuracy of the model responses was assessed against human expert annotations. While human experts determined whether an event required action, the models were instructed to assign a priority level on a predefined scale: zero, low, medium, high, and critical. This priority served as a tunable threshold for decision-making. The action threshold was set for evaluation purposes at either greater than medium or equal to or greater than medium, depending on the decision outcome. Since the models were not explicitly trained on event severity, the priority threshold is proposed as a hyperparameter, allowing for flexibility in practical deployment.

The standard machine learning metrics were applied to measure the success of model decisions, specifically, whether an event should be discarded or investigated (human expert decisions compared to model responses). Since the dataset is imbalanced (some False Positive alerts were added), the F1-score was chosen as the primary performance metric alongside precision, which emphasizes false positive predictions. Precision is particularly important in this context, as reducing false positives is critical to preventing unnecessary alerts and operator fatigue.

While the decision accuracy of the models was quantitatively evaluated in the first two demonstration cases using these metrics, the third case—assessing the quality of model-generated explanations—required a manual review. Each model’s responses were analyzed to determine their alignment with human expert interpretations, noting key strengths and

²https://github.com/koumajos/explainable_anomaly_detection_using_LLM

Table 6.3: Results of Case 1: LLM as Autonomous Anomaly Analyst Agent

Model	F1	Precision	Priority
GPT-4	0.81	0.92	$> \textit{medium}$
GPT-4o	0.86	0.92	$> \textit{medium}$
GPT-4o-mini	0.75	0.81	$\geq \textit{medium}$
Gemini 2.0 Flash Experimental	0.92	0.93	$> \textit{medium}$

limitations in explanation depth and relevance. The results as well as annotations are also available on GitHub².

Experimental Evaluation

In this section, three distinct use cases are demonstrated where LLMs assist in anomaly evaluation, prioritization, and explanation, thus enhancing the utility of anomaly detection systems in network security.

Case 1: LLM as Autonomous Anomaly Analyst Agent

In this scenario, the LLM acts as an autonomous analyst, evaluating outlier events identified by anomaly detection models. While anomalies may be statistically significant in a time-series context, they may not always indicate real network threats. Thus, the LLM is tasked with distinguishing genuine incidents from benign deviations.

To enable contextual reasoning, the LLM is provided with statistical summaries (mean and standard deviation) from 48 hours and seven days before and after the detected anomaly. This enables the model to analyze historical patterns, validate the anomaly’s significance, and determine whether it represents a false positive or a genuine threat (e.g., an attack or network disruption).

As seen in Table 6.3, the evaluated LLMs demonstrated a capability to prioritize network anomalies effectively. The Gemini 2.0 Flash Experimental model produced decisions most aligned with human expert assessments, followed by GPT-4o and GPT-4. The lightweight GPT-4o-mini exhibited slightly weaker performance, which is expected given its lower complexity.

Upon closer examination of discrepancies between model outputs and human evaluations, two primary cases of mislabeling compared to the human experts emerged. The first involved ambiguous scenarios where the significance of an event was subjective, often dependent on specific prompt phrasing (e.g., a model assigning medium priority, but could be higher priority and vice versa). The second involved differences in priority assignment, where all models correctly identified an anomaly but assigned varying levels of urgency. Notably, GPT-4o-mini tended to assign slightly lower priority compared to the others.

Overall, even the lightweight GPT-4o-mini provided competent results, demonstrating that LLMs can act as effective anomaly analysts even in a zero-shot setting.

Table 6.4: Results of Case 2: LLM as Autonomous Anomaly Alerter Agent

Model	F1	Precision	Priority
GPT-4	0.92	0.88	$\geq$ <i>medium</i>
GPT-4o	0.89	0.96	$>$ <i>medium</i>
GPT-4o-mini	0.81	0.83	$\geq$ <i>medium</i>
Gemini 2.0 Flash Experimental	0.93	0.96	$>$ <i>medium</i>

Case 2: LLM as Autonomous Anomaly Alerter Agent

In this use case, the LLM functions as a direct decision-maker, superseding the network operator by determining which detected anomalies warrant further investigation. Unlike the previous case, the model is only given context from 48 hours and seven days preceding the event (mean and standard deviation), simulating real-time anomaly detection without access to future observations. This setup assesses the LLM’s ability to efficiently filter out false positives.

Table 6.4 highlights that all evaluated models effectively filtered false positives, even without post-event context. The Gemini 2.0 Flash Experimental model once again exhibited the highest agreement with human experts, followed closely by GPT-4 and GPT-4o. While GPT-4o-mini displayed slightly lower precision, it remained a viable candidate for real-time filtering.

Interestingly, the absence of post-event context did not significantly degrade model performance. In some cases, models benefited from reduced cognitive overload, allowing them to focus on essential signals. Misprioritization primarily occurred at decision boundaries, reinforcing the need for a calibrated threshold for operational deployment. The results indicate that LLMs can autonomously assist in real-time alert triaging, reducing the burden on human analysts.

Case 3: LLM as a Human Analyst Companion

Beyond autonomous decision-making, LLMs can also serve as on-demand assistants, helping analysts interpret and contextualize network anomalies. This scenario evaluates the models’ ability to provide accurate and actionable explanations for detected anomalies, aiding less experienced operators.

Manual analysis of model-generated explanations revealed that:

- GPT-4 provided mostly accurate attack identifications but occasionally underestimated the severity of certain incidents.
- GPT-4o produced the most detailed and actionable explanations, closely matching human expert assessments.
- Gemini 2.0 Flash Experimental also performed well, aligning closely with GPT-4o in explaining anomalous network behavior.

- GPT-4o-mini, while capable of describing network patterns, struggled with deeper insights and concrete attack attributions.

When post-event context was included, models did not consistently improve. In fact, GPT-4 shifted towards a more conservative stance, reducing its confidence in attack classifications. GPT-4o also exhibited slight conservatism but maintained strong recommendations for operator intervention. Conversely, Gemini 2.0 Flash Experimental refined certain explanations to better match human assessments.

Overall, the LLMs demonstrated their capability to autonomously generate meaningful anomaly descriptions, making them valuable companions for network security teams.

Discussion of LLM Usability

The experiments demonstrate that LLMs can be highly effective for analyzing network anomalies, even without prior training on specific network traffic events. The models were able to work with detected anomalies data, provide meaningful explanations, and help prioritize alerts, which can significantly assist network operators.

An interesting observation is that different models assigned different levels of priority to the same anomalies. This means that while a model was consistent in how it classified anomalies, its decisions sometimes differed from those of other models. This is similar to how human experts do not always agree on the severity of a network event. Because of this, it was suggested that the priority level assigned by an LLM should be used as an adjustable setting. Depending on their security needs, organizations could decide how sensitive they want the model to be when prioritizing alerts.

The results were evaluated both automatically and manually by human experts, with only a limited portion of events from the dataset analyzed due to the demanding nature of manual review. The models' outputs were compared to prior event labels to assess their alignment with human judgment, acknowledging that some degree of subjectivity and experience-based bias may be present.

Overall, the results indicate that all tested models performed well, with some differences in how they evaluated network anomalies. The Gemini 2.0 Flash Experimental and GPT-4o models were the most reliable in both explaining anomalies and assigning appropriate priority levels. The smaller GPT-4o-mini model, while still useful, was less effective in detecting security-related patterns and often provided only general descriptions of network activity, rather than detailed security insights.

This confirms that LLMs can be a valuable extension for network anomaly detection, offering promising improvements in explainability and alert prioritization, but there is still room for enhancement. Future work could focus on fine-tuning domain-specific models for cybersecurity tasks to boost accuracy and reliability, while also optimizing prompt engineering techniques and evaluating various LLM architectures. Another important direction is to explore open-source models and assess whether they can deliver comparable performance with greater transparency and flexibility. Additionally, integrating more contextual data sources—such as network topology, threat intelligence feeds, and historical

attack patterns—could further refine explanation and prioritization capabilities. Finally, deploying these models in different real-world settings, where human analysts interact with them, would provide deeper insights into their practical strengths and limitations.

6.7 Key Findings and Contributions

This chapter presents research on network threat detection using *Aggregated time series* from network traffic. Threats are detected using deployable and explainable anomaly detection. The key findings in this chapter are:

- In the network traffic monitoring domain, there is a lack of a reference dataset as a crucial obstacle to the evaluation of novel forecasting and anomaly detection methods.
- The novel, unique real-world dataset of aggregated time series called CESNET Time-Series24 was published to address this crucial obstacle in the evaluation of forecasting and anomaly detection.
- In network traffic, several types of anomalies were identified, which could be beneficial to detect for both management and security reasons.
- A system for anomaly detection using several forecasting-based methods was proposed and deployed in the ISP network.
- Forecasting-based models for anomaly detection typically generate numerous *False positives* during the initial phase and on sparse time series. Therefore, sending out each detection anomaly will result in spamming the SoC team. Nevertheless, the SoC team would disable the detection or filtering of incidents. Therefore, the system alerts only to critical incidents, and the rest of the incidents are logged in the system for possible further evaluation if needed.
- The anomaly detection system shows multiple outliers that can represent various types of anomalies occurring on the network. While experienced users of such a system can quickly identify the type of issue, regular users may struggle. The system might benefit from automatic explanation and categorization of different types of anomalies.
- The LLM was successfully used for the explanation of detected anomalies. The LLM was evaluated on real-world data in multiple case studies to find all possibilities in using LLM for anomaly detection in network traffic.
- The approach demonstrates LLMs potential in *False positive* reduction, autonomous decision-making, and improved explainability for network analysts.

The results in this chapter indicate a significant advancement in monitoring encrypted and automated network traffic. By applying Time Series Analysis (TSA) to *Aggregated*

time series, security experts can investigate the abnormalities occurring in network traffic. The LLM demonstrates the potential in *False positive* reduction, autonomous decision-making, and improved explainability for network analysts. The developed methods offer a novel approach to modern threat detection in high-speed environments.

Conclusions

Visibility into network traffic for security reasons is limited due to the widespread adoption of encryption. Therefore, detecting security threats has become increasingly challenging in recent years. It is crucial to identify suitable algorithms and mathematical methods for detecting security threats in high-speed network traffic using side-channel analysis. Time Series Analysis (TSA) is crucial for modeling network traffic and detection of security threats, because time series are a natural representation of network traffic.

7.1 Summary

This dissertation thesis shows significant usability of TSA in detecting security threats in network traffic. The time series serves as a side channel of encryption, which can be leveraged using TSA in combination with Machine Learning (ML) or Deep Learning (DL). This dissertation thesis analysed multiple time series types that are possible to create from network traffic. Moreover, all proposed time series types were successfully utilized for detecting security threats in network traffic using novel methods proposed in the thesis author's publications. Furthermore, all proposed methods also focus on deployability into real-world high-speed networks.

In this dissertation thesis, *Single flow time series* were defined, which were successfully used for network traffic classification and detection of security threats. The novel approach mines time-series-based features from *Single flow time series*. The feature set is universally applicable to numerous tasks in the network traffic classification domain and matches State of the Art (SOTA) accuracy.

Moreover, the *Flow time series* were proposed with a creation process based on *Network dependencies*. The periodicities in network traffic are clearly visible and detectable in *Flow time series*. A novel method for detecting periodicity in network traffic was proposed, accompanied by a set of features that describe the behavior of periodicity. The feature set was successfully utilized for classifying applications, services, and operating systems, as well as detecting cryptominers and Command and Control (C&C) of botnets.

Furthermore, the *Aggregated time series* were utilized for threat detection using an anomaly detection. The novel, unique, real-world dataset CESNET TimeSeries24 was published to the scientific community. The system for anomaly detection was proposed and evaluated in a real-world Internet Service Provider (ISP) network. Detected anomalies were further explained by proposing a novel approach using Large Language Models (LLM).

All methods proposed in this thesis were successfully published by the thesis author and his colleagues. The thesis author's publications were presented and discussed within the scientific community. Together, the results were presented at five conferences and in two journals (both Q1 journals, according to SCImago). Alongside the papers, datasets were also published addressing open-science and enabling further research for the scientific community (9 datasets on Zenodo and one on GitHub). Moreover, alongside the papers, source codes of experience were also published, addressing open-science and reproducible research.

7.2 Contributions of the Dissertation Thesis

1. A universal TSA feature set was shown to outperform or achieve the same performance as related works in most binary and multiclass classification tasks in the domain of network traffic classification. The method was successfully applied to these tasks: classification and detection of Botnets, Cryptomining, DNS malware, DoH, DoS, HTTPS Brute Force, Intrusion detection, IoT malware, TOR, and VPN.
2. The deployable flow extension called NetTiSA was proposed, which is a reduced universal TSA feature set with similar performance.
3. A novel approach for utilizing *Flow time series* and their periodic behavior was proposed by using the Lomb-Scargle periodogram. The approach is applicable for multiple case studies, including:
 - a) multiclass classification of applications, services, and operating systems achieved the highest performance with 91% Accuracy and a 90% F1-score
 - b) cryptomining detection achieving up to 95.19% Accuracy and a 92.21% F1-score
 - c) botnet C&C communication achieves 100% Precision across multiple datasets, making it a highly reliable tool for high-confidence threat detection
4. The novel dataset CESNET-CC25 for detecting C&C communication in botnets was proposed.
5. The novel, unique real-world dataset of *Aggregated time series* called CESNET TimeSeries24 was published to address this crucial obstacle in the evaluation of forecasting and anomaly detection.
6. A system for anomaly detection using several forecasting-based methods applied on *Aggregated time series* was proposed and deployed in the ISP network.

7. The LLM was successfully utilized for explaining detected anomalies in real-world data across multiple case studies, exploring all possibilities for using LLMs for explainable anomaly detection in network traffic.

7.3 Future Work

While this dissertation thesis establishes a rigorous network time series analysis, several emerging paths can be further investigated to address the evolving landscape of network security. The thesis author suggests the following research directions, which lie beyond the current scope:

- Future research should investigate the integration of advanced DL architectures—such as Transformers or Temporal Convolutional Networks—specifically for forecasting-based anomaly detection. By modeling complex non-linear dependencies in network traffic, these models could improve the sensitivity of outlier detection in highly dynamic environments.
- There is a critical need for the creation of standardized, high-quality benchmarks for network forecasting and outlier detection. Establishing such benchmarks would facilitate objective comparisons between disparate methodologies and ensure that performance metrics are evaluated against consistent ground-truth scenarios.
- To enhance scientific transparency, future efforts should focus on developing open-source tools and platforms designed for the reproducible evaluation of network experiments. Such tools would streamline the validation of results across different research groups and hardware environments.
- The deployment of localized, privacy-preserving LLMs offers a promising direction for translating complex statistical anomalies into human-understandable explanations. Research on explainable traffic analysis could bridge the gap between automated detection systems and human security analysts.
- Parallel to the deployment of LLMs, a standardized benchmark must be established to evaluate the accuracy, fidelity, and safety of LLM-generated explanations concerning security threats and anomalous traffic patterns.
- Future work should explore the synergy between time-series-based methods and other analytical paradigms, such as graph-based analysis or semantic payload inspection. Information fusion techniques can provide a more comprehensive view of network state, thereby reducing false-positive rates in complex attack scenarios.
- The applicability of Large Foundation Models (pre-trained on vast quantities of cross-domain data) should be evaluated for zero-shot or few-shot network traffic classification tasks. Assessing their generalizability to domain-specific protocols remains a significant and open challenge.

Bibliography

- [1] Internet Protocol. RFC 791, September 1981. URL: <https://www.rfc-editor.org/info/rfc791>, doi:10.17487/RFC0791.
- [2] Amaia Abanda, Usue Mori, and Jose A. Lozano. A review on distance based time series classification. *Data Mining and Knowledge Discovery*, 33(2):378–412, 2019. doi:10.1007/s10618-018-0596-4.
- [3] Shubair A Abdullah and Amaal S Al-Hashmi. Tisefe: Time series evolving fuzzy engine for network traffic classification. *International Journal of Communication Networks and Information Security*, 10(1):116–124, 2018.
- [4] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescapé. Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE transactions on network and service management*, 16(2):445–458, 2019.
- [5] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescapé. Distiller: Encrypted traffic classification via multimodal multitask deep learning. *Journal of Network and Computer Applications*, 183:102985, 2021.
- [6] Giuseppe Aceto et al. DISTILLER: Encrypted Traffic Classification via Multimodal Multitask Deep Learning. *J. Netw. Comput. Appl.*, 2021. doi:10.1016/j.jnca.2021.102985.
- [7] Ratnadip Adhikari and Ramesh K Agrawal. An introductory study on time series modeling and forecasting, 2013.
- [8] Georgios Agraftotis et al. Image-based neural network models for malware traffic classification using pcap to picture conversion. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, pages 1–7, 2022.

- [9] Rakesh Agrawal et al. Fast Algorithms for Mining Association Rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499. Santiago, Chile, 1994.
- [10] Hana Ahmed et al. Managing randomness to enable reproducible machine learning. In *Proceedings of the 5th International Workshop on Practical Reproducible Evaluation of Computer Systems*, P-RECS '22, Jun 2022. doi:10.1145/3526062.3536353.
- [11] Iman Akbari, Mohammad A Salahuddin, Leni Ven, Noura Limam, Raouf Boutaba, Bertrand Mathieu, Stephanie Moteau, and Stephane Tuffin. Traffic classification in an increasingly encrypted web. *Communications of the ACM*, 65(10):75–83, 2022.
- [12] Mohammed M Alani. Botstop: Packet-based efficient and explainable iot botnet detection using machine learning. *Computer Communications*, 193:53–62, 2022.
- [13] Theyazn HH Aldhyani et al. Intelligent Hybrid Model to Enhance Time Series Models for Predicting Network Traffic. *IEEE Access*, 8:130431–130451, 2020.
- [14] Omar Alghushairy, Raed Alsini, Terence Soule, and Xiaogang Ma. A review of local outlier factor algorithms for outlier detection in big data streams.
- [15] Shadi Aljawarneh, Vangipuram Radhakrishna, Puligadda Veereswara Kumar, and Vinjamuri Janaki. A similarity measure for temporal pattern discovery in time series data generated by iot. In *2016 International conference on engineering & MIS (ICEMIS)*, pages 1–4. IEEE, 2016.
- [16] Blake Anderson and David McGrew. Identifying encrypted malware traffic with contextual flow data. In *Proceedings of the 2016 ACM workshop on artificial intelligence and security*, pages 35–46, 2016.
- [17] Tomasz Andrysiak, Łukasz Saganowski, Michał Choraś, and Rafał Kozik. Network traffic prediction and anomaly detection based on arfima model. In *International Joint Conference SOCO'14-CISIS'14-ICEUTE'14: Bilbao, Spain, June 25th-27th, 2014, Proceedings*, pages 545–554. Springer, 2014.
- [18] Azeem Aqil, Karim Khalil, Ahmed O.F. Atya, Evangelos E. Papalexakis, Srikanth V. Krishnamurthy, Trent Jaeger, K. K. Ramakrishnan, Paul Yu, and Ananthram Swami. Jaal: Towards network intrusion detection at isp scale. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '17*, page 134–146, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3143361.3143399.
- [19] Goethem Arthur Van, Frank Staals, Maarten Löffler, Jason Dykes, and Bettina Speckmann. Multi-granular trend detection for time-series analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):661–670, 2017. doi:10.1109/TVCG.2016.2598619.

-
- [20] Basil AsSadhan, Jose M. F. Moura, and David Lapsley. Periodic behavior in botnet command and control channels traffic. In *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, pages 1–6, 2009. doi:10.1109/GLOCOM.2009.5426172.
- [21] Basil AsSadhan, Jose MF Moura, and David Lapsley. Periodic behavior in botnet command and control channels traffic. In *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*, pages 1–6. IEEE, 2009.
- [22] D. Astakhov, V. Mazurenko, A. Fedorovych, and N. Lysenko. Methods for detecting statistical changes in network traffic characteristics.
- [23] Lei Bai, Lina Yao, Salil S. Kanhere, Xianzhi Wang, and Zheng Yang. Automatic device classification from network traffic streams of internet of things. In *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pages 1–9, 2018. doi:10.1109/LCN.2018.8638232.
- [24] Kurt Barbé et al. Welch Method Revisited: Nonparametric Power Spectrum Estimation Via Circular Overlap. *IEEE Trans. Signal Process.*, 58(2):553–565, 2010. doi:10.1109/TSP.2009.2031724.
- [25] M. S. Bartlett. Periodogram Analysis and Continuous Spectra, Jun 1950. URL: <https://academic.oup.com/biomet/article/37/1-2/1/194725>.
- [26] Vasiliki A Basdekidou. The momentum & trend-reversal as temporal market anomalies. *International Journal of Economics and Finance*, 9(5):1–19, 2017.
- [27] Matthew Behnke et al. Feature engineering and machine learning model comparison for malicious activity detection in the dns-over-https protocol. *IEEE Access*, 9:129902–129916, 2021.
- [28] Tomas Benes, Jaroslav Pesek, and Tomas Cejka. Look at my network: An insight into the isp backbone traffic. In *2023 19th International Conference on Network and Service Management (CNSM)*, pages 1–7. IEEE, 2023.
- [29] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013.
- [30] Jarosław Bernacki and Grzegorz Kołaczek. Anomaly detection in network traffic using selected methods of time series analysis. *International Journal of Computer Network and Information Security*, 2015.
- [31] Alina Beygelzimer, Emre Erdogan, Sheng Ma, and Irina Rish. Statistical models for unequally spaced time series. 04 2005. doi:10.1137/1.9781611972757.74.

- [32] Monowar H Bhuyan, Dhruva Kr Bhattacharyya, and Jugal K Kalita. Surveying port scans and their detection methodologies. *The Computer Journal*, 54(10):1565–1581, 2011.
- [33] Jing Bi, Xiang Zhang, Haitao Yuan, Jia Zhang, and MengChu Zhou. A hybrid prediction method for realistic network traffic with temporal convolutional network and lstm. *IEEE Transactions on Automation Science and Engineering*, 19(3):1869–1879, 2021.
- [34] Thitima Booranawong and Apidet Booranawong. Double exponential smoothing and holt-winters methods with optimal initial values and weighting factors for forecasting lime, thai chili and lemongrass prices in thailand. *Engineering and Applied Science Research*, 45:32–38, 02 2018. doi:10.14456/easr.2018.5.
- [35] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- [36] E Oran Brigham. *The fast Fourier transform and its applications*. Prentice-Hall, Inc., 1988.
- [37] Sherree Buchenroth and Robert Jennings. A descriptive analysis of the time series behavior of financial analysts’ earnings forecasts. *Quarterly Journal of Business and Economics*, 26(3):22–41, 1987. URL: <http://www.jstor.org/stable/40472880>.
- [38] Tomas Cejka, Vaclav Bartos, Marek Svepes, Zdenek Rosa, and Hana Kubatova. Nemea: a framework for network traffic analysis. In *CNSM*, 2016.
- [39] Merve Çelebi, Alper Özbilen, and Uraz Yavanoğlu. A comprehensive survey on deep packet inspection for advanced network traffic analysis: issues and challenges. *Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi*, 12(1):1–29, 2023.
- [40] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.
- [41] Chris Chatfield and Haipeng Xing. *The analysis of time series: an introduction with R*. Chapman and hall/CRC, 2019.
- [42] Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, et al. Automatic root cause analysis via large language models for cloud incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems*, pages 674–688, 2024.
- [43] Zhitang Chen et al. Predicting future traffic using hidden markov models. In *IEEE ICNP*, 2016.

-
- [44] Zhitang Chen, Ke He, Jian Li, and Yanhui Geng. Seq2img: A sequence-to-image based approach towards ip traffic classification using convolutional neural networks. In *2017 IEEE International conference on big data (big data)*, pages 1271–1276. IEEE, 2017.
 - [45] Seshadri Rao Chinta, Vinod Babu Polinati, and PN Srinivas. Detecting bots inside a host using network behavior analysis. *International Journal of Computer Applications*, 975:8887, 2018.
 - [46] Kenjiro Cho, Koushirou Mitsuya, and Akira Kato. Traffic data repository at the {WIDE} project. In *2000 USENIX Annual Technical Conference (USENIX ATC 00)*, 2000.
 - [47] Sudipta Chowdhury et al. Botnet Detection Using Graph-based Feature Clustering. *J. Big Data*, 4:14, 2017. doi:10.1186/s40537-017-0074-7.
 - [48] Benoit Claise et al. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. *RFC*, 7011:1–76, 2013. doi:10.17487/RFC7011.
 - [49] Michael P Clements et al. On the limitations of comparing mean square forecast errors. *Journal of Forecasting*, 12(8), 1993.
 - [50] G. Creech and J. Hu. *ADFA IDS Dataset, University of Arizona Artificial Intelligence Lab, AZSecure-data*. Director Hsinchun Chen, November 2016. <http://www.azsecure-data.org/>.
 - [51] Juan C Cuevas-Tello, Peter Tiño, Somak Raychaudhury, Xin Yao, and Markus Harva. Uncovering delayed patterns in noisy and irregularly sampled time series: an astronomy application. *Pattern Recognition*, 43(3):1165–1179, 2010.
 - [52] Alfredo Cuzzocrea et al. Tor Traffic Analysis and Detection via Machine Learning Techniques. In *IEEE BigData 2017*, 2017. doi:10.1109/BigData.2017.8258487.
 - [53] Jianbang Dai et al. Glads: A global-local attention data selection model for multimodal multitask encrypted traffic classification of iot. *Computer Networks*, 225:109652, 2023.
 - [54] Anestis Dalgkitsis et al. Traffic forecasting in cellular networks using the lstm rnn. In *PCI*, 2018.
 - [55] Murat Dener et al. Rfse-gru: Data balanced classification model for mobile encrypted traffic in big data environment. *IEEE Access*, 11:21831–21847, 2023.
 - [56] Hongwei Ding et al. Imbalanced data classification: A knn and generative adversarial networks-based hybrid approach for intrusion detection. *Future Generation Computer Systems*, 131:240–254, 2022.

- [57] Gerard Draper-Gil et al. Characterization of Encrypted and VPN Traffic Using Time-related. In *ICISSP*, pages 407–414, 2016.
- [58] Pierre Duchesne and Maria Pacurar. Evaluating financial time series models for irregularly spaced data: a spectral density approach. *Computers & operations research*, 35(1):130–155, 2008.
- [59] Alessandro D’Alconzo, Idilio Drago, Andrea Morichetta, Marco Mellia, and Pedro Casas. A survey on big data for network traffic monitoring and analysis. *IEEE Transactions on Network and Service Management*, 16(3):800–813, 2019.
- [60] Mohamed G Elfeky, Walid G Aref, and Ahmed K Elmagarmid. Periodicity detection in time series databases. *IEEE Transactions on Knowledge and Data Engineering*, 17(7):875–887, 2005.
- [61] Nelly Elsayed et al. Deep gated recurrent and convolutional network hybrid model for univariate time series classification. *arXiv preprint arXiv:1812.07683*, 2018.
- [62] Meisam Eslahi, M. S. Rohmad, Hamid Nilsaz, Maryam Var Naseri, N.M. Tahir, and H. Hashim. Periodicity classification of HTTP traffic to detect HTTP Botnets. In *2015 IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE)*, pages 119–123, 2015. doi:10.1109/ISCAIE.2015.7298339.
- [63] Marzieh Fathi, Mostafa Haghi Kashani, Seyed Mahdi Jameii, and Ebrahim Mahdipour. Big data analytics in weather forecasting: A systematic review. *Archives of Computational Methods in Engineering*, 29(2):1247–1275, 2022.
- [64] Gilberto Fernandes, Joel JPC Rodrigues, Luiz Fernando Carvalho, Jalal F Al-Muhtadi, and Mario Lemes Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, 70:447–489, 2019.
- [65] Mohamed Amine Ferrag et al. Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications: Centralized and federated learning, 2022. doi:10.21227/mbc1-1h68.
- [66] Gabriel O Ferreira, Chiara Ravazzi, Fabrizio Dabbene, Giuseppe C Calafiore, and Marco Fiore. Forecasting network traffic: A survey and tutorial with open-source comparative evaluation. *IEEE Access*, 11:6018–6044, 2023.
- [67] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. MAW-ILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In Jaudelice Cavalcante de Oliveira, Maximilian Ott, Timothy G. Griffin, and Muriel Médard, editors, *Proceedings of the 2010 ACM Conference on Emerging Networking Experiments and Technology, CoNEXT 2010, Philadelphia, PA, USA, November 30 - December 03, 2010*, page 8. ACM, 2010. doi:10.1145/1921168.1921179.

-
- [68] Student Union for Electrical Engineering (Fachbereichsvertretung Elektrotechnik) at Ulm University and Philipp Hinz. *2017 SUEE Data Set*. <https://github.com/vs-uulm/2017-SUEE-data-set>, Accessed: 2022-06-24.
- [69] Linux Foundation. Data plane development kit (DPDK), 2015. URL: <http://www.dpdk.org>.
- [70] Fabio Frescura et al. Significance Tests for Periodogram Peaks. 2007. URL: https://www.researchgate.net/publication/1894545_Significance_Tests_for_Periodogram_Peaks.
- [71] Sebastian Garcia et al. IoT-23: A labeled dataset with malicious and benign IoT network traffic, January 2020. More details here <https://www.stratosphereips.org/datasets-iot23>. doi:10.5281/zenodo.4743746.
- [72] Sebastian Garcia, Martin Grill, Jan Stiborek, and Alejandro Zunino. An empirical comparison of botnet detection methods. *computers & security*, 45:100–123, 2014.
- [73] Sebastián García, Karel Hynek, Dmitrii Vekshin, Tomáš Čejka, and Armin Wasicsek. Large scale measurement on the adoption of encrypted dns. *arXiv preprint arXiv:2107.04436*, 2021.
- [74] S. García et al. An Empirical Comparison of Botnet Detection Methods. *Computers & Security*, 45:100–123, 2014. doi:10.1016/j.cose.2014.05.011.
- [75] F Godtliebsen, LR Olsen, and J-G Winther. Recent developments in statistical time series analysis: Examples of use in climate research. *Geophysical Research Letters*, 30(12), 2003.
- [76] Guofei Gu, Junjie Zhang, and Wenke Lee. Botsniffer: Detecting botnet command and control channels in network traffic. In *15th Annual Network and Distributed System Security Symposium*, 2008.
- [77] Jorge Luis Guerra, Carlos Catania, and Eduardo Veas. Datasets are not enough: Challenges in labeling network traffic. *Computers & Security*.
- [78] Dani Gunawan, Tika Hairani, and Ainul Hizriadi. Botnet identification based on flow traffic by using k-nearest neighbor. In *2019 International Conference on Advanced Computer Science and information Systems (ICACSIS)*, pages 95–100. IEEE, 2019.
- [79] Ralph Haberl et al. Comparison of Frequency and Time Domain Analysis of the Signal-averaged Electrocardiogram in Patients With Ventricular Tachycardia and Coronary Artery Disease: Methodologic Validation and Clinical Relevance. *Journal of the American College of Cardiology*, 12(1):150–158, 1988. doi:10.1016/0735-1097(88)90368-3.

- [80] Mackenzie Haffey et al. Modeling, Analysis, and Characterization of Periodic Traffic on a Campus Edge Network. In *26th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS 2018, Milwaukee, WI, USA, September 25-28, 2018*, pages 170–182. IEEE Computer Society, 2018. doi:10.1109/MASCOTS.2018.00025.
- [81] James D Hamilton. *Time series analysis*. Princeton university press, 2020.
- [82] Sarah Harun et al. Bot Classification for Real-life Highly Class-imbalanced Dataset. In *DASC/PiCom/DataCom/CyberSciTech 2017*, pages 565–572. IEEE Computer Society, 2017. doi:10.1109/DASC-PiCom-DataCom-CyberSciTec.2017.102.
- [83] Emad S Hassan et al. Optimizing bandwidth utilization and traffic control in isp networks for enhanced smart agriculture. *Plos one*, 19, 2024.
- [84] Shital P Hatkar, SH Kadam, and AH Syed. Analysis of various periodicity detection algorithms in time series data with design of new algorithm. *International Journal of Computer Applications Technology and Research*, 3:229–239, 2014.
- [85] D. M. Hawkins. *Identification of Outliers*. Springer Netherlands.
- [86] Kaiming He et al. Deep residual learning for image recognition. In *IEEE CVPR*, 2016.
- [87] Manuel Herrera, Manu Sasidharan, Jorge Merino, and Ajith K Parlikad. Handling irregularly sampled iot time series to inform infrastructure asset management. *IFAC-PapersOnLine*, 55(19):241–245, 2022.
- [88] Paul E. Hoffman and Patrick McManus. DNS Queries over HTTPS (DoH). RFC 8484, October 2018. URL: <https://www.rfc-editor.org/info/rfc8484>, doi:10.17487/RFC8484.
- [89] Rick Hofstede, Mattijs Jonker, Anna Sperotto, and Aiko Pras. Flow-based web application brute-force attack and compromise detection. *Journal of Network and Systems Management*, August 2017. doi:10.1007/s10922-017-9421-4.
- [90] Rick Hofstede, Pavel Čeleda, Brian Trammell, Idilio Drago, Ramin Sadre, Anna Sperotto, and Aiko Pras. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Communications Surveys & Tutorials*, 16(4):2037–2064, 2014. doi:10.1109/COMST.2014.2321898.
- [91] Geng Hong et al. How you get shot in the back: A systematical study about crypto-jacking in the real world. In *ACM SIGSAC*, 2018. doi:10.1145/3243734.3243840.
- [92] James H Horne et al. A Prescription for Period Analysis of Unevenly Sampled Time Series. *The Astrophysical Journal*, 302:757–763, 1986.

-
- [93] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul E. Hoffman. Specification for DNS over Transport Layer Security (TLS). RFC 7858, May 2016. URL: <https://www.rfc-editor.org/info/rfc7858>, doi:10.17487/RFC7858.
- [94] Min Hur et al. Towards smart phone traffic classification. In *APNOMS*, pages 1–4, 2012. doi:10.1109/APNOMS.2012.6356064.
- [95] Martin Husák et al. Security Monitoring of HTTP Traffic Using Extended Flows. In *ARES 2015*. IEEE Computer Society. doi:10.1109/ARES.2015.42.
- [96] Martin Husák, Jana Komárková, Elias Bou-Harb, and Pavel Čeleda. Survey of attack projection, prediction, and forecasting in cyber security. *IEEE Communications Surveys & Tutorials*, 21(1):640–660, 2018.
- [97] Ren-Hung Hwang, Min-Chun Peng, Chien-Wei Huang, Po-Ching Lin, and Van-Linh Nguyen. An unsupervised deep learning model for early network traffic anomaly detection. *IEEE Access*, 8:30387–30399, 2020. doi:10.1109/ACCESS.2020.2973023.
- [98] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.
- [99] Karel Hynek, Tomáš Beneš, Tomáš Čejka, and Hana Kubátová. Refined detection of ssh brute-force attackers using machine learning. In *ICT Systems Security and Privacy Protection: 35th IFIP TC 11 International Conference, SEC 2020, Maribor, Slovenia, September 21–23, 2020, Proceedings 35*, pages 49–63. Springer, 2020.
- [100] Wan Nur Hidayah Ibrahim, Syahid Anuar, Ali Selamat, Ondrej Krejcar, Rubén González Crespo, Enrique Herrera-Viedma, and Hamido Fujita. Multilayer framework for botnet detection using machine learning algorithms. *IEEE Access*, 9:48753–48768, 2021.
- [101] Hassan Ismail Fawaz et al. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 2020.
- [102] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.
- [103] Racquel Jandoc, Andrea M Burden, Muhammad Mamdani, Linda E Lévesque, and Suzanne M Cadarette. Interrupted time series analysis in drug utilization research is increasing: systematic review and recommendations. *Journal of clinical epidemiology*, 68(8):950–956, 2015.
- [104] Kamil Jerabek, Karel Hynek, Ondrej Rysavy, and Ivana Burgetova. Dns over https detection using standard flow telemetry. *IEEE Access*, 11:50000–50012, 2023. doi:10.1109/ACCESS.2023.3275744.

- [105] Kamil Jerabek, Jan Luxemburk, Richard Plny, Josef Koumar, Jaroslav Pesek, and Karel Hynek. When simple model just works: Is network traffic classification in crisis? *arXiv preprint arXiv:2506.08655*, 2025.
- [106] Kamil Jeřábek et al. Collection of datasets with DNS over HTTPS traffic. *Data in Brief*, 42:108310, 2022. doi:10.1016/j.dib.2022.108310.
- [107] Jianguo Jiang, Qilei Yin, Zhixin Shi, Meimei Li, and Bin Lv. A new c&c channel detection framework using heuristic rule and transfer learning. In *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, pages 1–9. IEEE, 2019.
- [108] Ming JIANG, Chun-ming WU, Min Zhang, and Da-min HU. Research on the comparison of time series models for network traffic prediction. *ACTA ELECTRONICA SINICA*, 37(11):2353, 2009.
- [109] Steven Jorgensen et al. Extensible Machine Learning for Encrypted Network Traffic Application Labeling via Uncertainty Quantification. *CoRR*, abs/2205.05628, 2022. arXiv:2205.05628, doi:10.48550/arXiv.2205.05628.
- [110] Steven Jorgensen et al. Extensible Machine Learning for Encrypted Network Traffic Application Labeling via Uncertainty Quantification. *CoRR*, abs/2205.05628, 2022. arXiv:2205.05628, doi:10.48550/arXiv.2205.05628.
- [111] Manish Joshi et al. A review of network traffic analysis and prediction techniques. *arXiv preprint arXiv:1507.05722*, 2015.
- [112] Sangjoon Jung et al. A Prediction Method of Network Traffic Using Time Series Models. In *ICCSA 2006*, volume 3982, pages 234–243. Springer, 2006. doi:10.1007/11751595_26.
- [113] Huang Kai et al. Network Anomaly Detection Based on Statistical Approach and Time Series Analysis. In *2009 International Conference on Advanced Information Networking and Applications Workshops*, pages 205–211. IEEE, 2009.
- [114] Fazle Karim et al. Lstm fully convolutional networks for time series classification. *IEEE access*, 6:1662–1669, 2017.
- [115] Fazle Karim et al. Insights into lstm fully convolutional networks for time series classification. *Ieee Access*, 7, 2019.
- [116] Rezaul Karim, Md. Atiqul Islam Rizvi, and Mohammad Shamsul Arefin. A survey on anomaly detection strategies. In *Second International Conference on Image Processing and Capsule Networks*.
- [117] Juan Pablo Karmy and Sebastián Maldonado. Hierarchical time series forecasting via support vector regression in the european travel retail industry. *Expert Systems with Applications*, 137:59–73, 2019.

-
- [118] Amina Khacha et al. Hybrid deep learning-based intrusion detection system for industrial internet of things. In *ISIA*. IEEE, 2022.
- [119] Mehdi Khashei et al. Which methodology is better for combining linear and nonlinear models for time series forecasting? *Journal of Industrial and Systems Engineering*, 4(4):265–285, 2011.
- [120] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *Advances in neural information processing systems*, 33:6696–6707, 2020.
- [121] Kyoung-jae Kim. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1-2):307–319, 2003.
- [122] Tomáš Komárek, Jan Brabec, Čeněk Škarda, and Petr Somol. Threat hunting as a similarity search problem on multi-positive and unlabeled data. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 2098–2103. URL: <https://ieeexplore.ieee.org/document/9671958>, doi:10.1109/BigData52589.2021.9671958.
- [123] Qian Kong, Xu Zhang, Chongfu Zhang, Limengnan Zhou, Miao Yu, Yutong He, Youxian Chen, Yu Miao, and Haijun Yuan. Network traffic prediction: Apply the transformer to time series forecasting. *Mathematical Problems in Engineering*, 2022(1):8424398, 2022.
- [124] Abdelaziz Amara Korba, Aledine Diaf, and Yacine Ghamri-Doudane. Ai-driven fast and early detection of iot botnet threats: A comprehensive network traffic analysis approach. In *2024 International Wireless Communications and Mobile Computing (IWCMC)*, pages 1779–1784. IEEE, 2024.
- [125] Nickolaos Koroniotis et al. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Gener. Comput. Syst.*, 100:779–796, 2019. doi:10.1016/j.future.2019.05.041.
- [126] Joos Korstanje. *Advanced Forecasting with Python: With State-of-the-Art-Models Including LSTMs, Facebook’s Prophet, and Amazon’s DeepAR*.
- [127] Georgios Kougioumtzidis, Vladimir K Poulkov, Pavlos I Lazaridis, and Zaharias D Zaharis. Mobile network traffic prediction using temporal fusion transformer. *IEEE Transactions on Artificial Intelligence*, 2025.
- [128] Peter Kromkowski et al. Evaluating Statistical Models for Network Traffic Anomaly Detection. In *2019 systems and information engineering design symposium (SIEDS)*, pages 1–6. IEEE, 2019.
- [129] M Kumaar et al. A Hybrid Framework for Intrusion Detection in Healthcare Systems Using Deep Learning. *Frontiers in Public Health*, 9, 2021.

- [130] R. Kumari, Sheetanshu, M. K. Singh, R. Jha, and N.K. Singh. Anomaly detection in network traffic using k-mean clustering. In *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, pages 387–393, 2016. doi: 10.1109/RAIT.2016.7507933.
- [131] Yesi Novaria Kunang, Siti Nurmaini, Deris Stiawan, and Bhakti Yudho Suprpto. Attack classification of an intrusion detection system using deep learning and hyperparameter optimization. *Journal of Information Security and Applications*, 58:102804, 2021.
- [132] Arash Habibi Lashkari et al. Characterization of Tor Traffic using Time based Features. In *ICISSP 2017*, pages 253–262. SciTePress, 2017. doi:10.5220/0006105602530262.
- [133] Sihyung Lee, Kyriaki Levanti, and Hyong S Kim. Network monitoring: Present and future. *Computer Networks*, pages 84–98, 2014.
- [134] Miguel Leon, Tijana Markovic, and Sasikumar Punnekkat. Comparative evaluation of machine learning algorithms for network intrusion detection and attack classification. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 01–08, 2022. doi:10.1109/IJCNN55064.2022.9892293.
- [135] Alexander Lerch. *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. Wiley-IEEE Press, 2012.
- [136] Ming Li, Qiang Li, Guangzhe Xuan, and Dong Guo. Identifying compromised hosts under apt using dns request sequences. *Journal of parallel and distributed computing*, 152:67–78, 2021.
- [137] Qinyong Li, Haoming Guo, Xinjian Shan, Zhixin Wang, Yanyan Wei, Jianxiu Bai, and Qihong Zhang. Swsd: An abnormal detection algorithm on unequally spaced time series for disaster prediction. In *Proceedings of the 2020 4th International Conference on Computer Science and Artificial Intelligence*, pages 219–224, 2020.
- [138] Ta-Hsin Li. Laplace Periodogram for Time Series Analysis. *Journal of the American Statistical Association*, 103(482):757–768, 2008.
- [139] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical transactions of the royal society a: mathematical, physical and engineering sciences*, 379(2194), 2021.
- [140] Jun Liu, Chaoyun Zhang, Jiaxu Qian, Minghua Ma, Si Qin, Chetan Bansal, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Large language models can deliver accurate and interpretable time series anomaly detection. *arXiv preprint arXiv:2405.15370*, 2024.

-
- [141] N.R. Lomb. Least-squares frequency analysis of unequally spaced data. 1976. doi:10.1007/BF00648343.
- [142] Manuel Lopez-Martin et al. Network traffic classifier with convolutional and recurrent neural networks for Internet of Things. *IEEE access*, 5:18042–18050, 2017.
- [143] Jan Luxemburk and Tomáš Čejka. Fine-grained tls services classification with reject option. *Computer Networks*, 220:109467, 2023.
- [144] Jan Luxemburk et al. Detection of https brute-force attacks with packet-level feature set. In *CCWC 2021*, pages 0114–0122, 2021. doi:10.1109/CCWC51732.2021.9375998.
- [145] Jan Luxemburk et al. Fine-grained TLS Services Classification With Reject Option. *Comput. Networks*, 220:109467, 2023. doi:10.1016/j.comnet.2022.109467.
- [146] Jan Luxemburk, Karel Hynek, and Tomas Cejka. HTTPS Brute-force dataset with extended network flows, November 2020. doi:10.5281/zenodo.4275775.
- [147] Jan Luxemburk, Karel Hynek, and Tomáš Čejka. Encrypted traffic classification: the quic case. In *2023 7th Network Traffic Measurement and Analysis Conference (TMA)*, pages 1–10. IEEE, 2023.
- [148] Minzhao Lyu, Hassan Habibi Gharakheili, and Vijay Sivaraman. A survey on enterprise network security: Asset behavioral monitoring and distributed attack detection. *IEEE Access*, 2024.
- [149] Qian Ma, Cong Sun, Baojiang Cui, and Xiaohui Jin. A novel model for anomaly detection in network traffic based on kernel support vector machine. *Computers & Security*, 104:102215, 2021. doi:10.1016/j.cose.2021.102215.
- [150] Zhen Ma, Qiang Li, and Xiangyu Meng. Discovering suspicious apt families through a large-scale domain graph in information-centric iot. *IEEE Access*, 7:13917–13926, 2019.
- [151] Rishabh Madan et al. Predicting Computer Network Traffic: A Time Series Forecasting Approach Using DWT, ARIMA and RNN. In *IC3 2018*, pages 1–5. IEEE Computer Society, 2018. doi:10.1109/IC3.2018.8530608.
- [152] Samaneh Mahdavifar et al. Classifying Malicious Domains using DNS Traffic Analysis. In *DASC/PiCom/CBDCoM/CyberSciTech 2021*, pages 60–67. IEEE, 2021. doi:10.1109/DASC-PiCom-CBDCoM-CyberSciTech52372.2021.00024.
- [153] Gonzalo Marín et al. Deep in the dark - deep learning-based malware traffic detection without expert knowledge. In *SPW 2019*, pages 36–42, 2019. doi:10.1109/SPW.2019.00019.

- [154] W.W. Melek, Ziren Lu, Alex Kapps, and William Fraser. Comparison of trend detection algorithms in the analysis of physiological time-series data. *IEEE transactions on bio-medical engineering*, 52:639–51, 05 2005. doi:10.1109/TBME.2005.844029.
- [155] Marcin Michalak, Łukasz Wawrowski, Marek Sikora, Rafał Kurianowicz, Artur Kozłowski, and Andrzej Białas. Outlier detection in network traffic monitoring. In *ICPRAM*.
- [156] MITRE. Groups. <https://attack.mitre.org/groups/>, n.d. Accessed: 2025-06-20.
- [157] H Zare Moayedi and MA Masnadi-Shirazi. Arima model for network traffic prediction and anomaly detection. In *2008 international symposium on information technology*, volume 4, pages 1–6. IEEE, 2008.
- [158] Mohammadreza MontazeriShatoori et al. Detection of DoH Tunnels using Time-series Classification of Encrypted Traffic. In *DASC/PiCom/CBDCCom/CyberSciTech 2020*, pages 63–70. IEEE, 2020. doi:10.1109/DASC-PiCom-CBDCCom-CyberSciTech49142.2020.00026.
- [159] Pablo Montero-Manso and Rob J Hyndman. Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*, 37(4):1632–1653, 2021.
- [160] Andrew Moore, Denis Zuev, and Crogan Michael. Discriminators for use in flow-based classification. *Queen Mary and Westfield College, Department of Computer Science*, 2005. URL: <https://www.cl.cam.ac.uk/~awm22/publications/moore2005discriminators.pdf>.
- [161] Mohammad Amin Morid, Olivia R Liu Sheng, and Joseph Dunbar. Time series prediction using deep learning methods in healthcare. *ACM Transactions on Management Information Systems*, 14(1):1–29, 2023.
- [162] Vassilios C Moussas et al. Network traffic modeling and prediction using multiplicative seasonal arima models. In *IC-EPSMSO*, pages 6–9, 2005.
- [163] 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.
- [164] 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.
- [165] Mohamed Naas et al. Encrypted VPN Dataset, November 2022. doi:10.5281/zenodo.7301756.

-
- [166] Satoshi Nakamoto. A peer-to-peer electronic cash system. *Bitcoin.org*, 4:2, 2008. URL: <https://bitcoin.org/bitcoin.pdf>.
- [167] Milad Nasr, Amir Houmansadr, and Arya Mazumdar. Compressive traffic analysis: A new paradigm for scalable traffic analysis. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17*, page 2053–2069, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3133956.3134074.
- [168] Brett Naul et al. A recurrent neural network for classification of unevenly sampled variable stars. *Nature Astronomy*, 2(2):151–155, 2018.
- [169] Mukrimah Nawir, Amiza Amir, Ong Bi Lynn, Naimah Yaakob, and R Badlishah Ahmad. Performances of machine learning algorithms for binary classification of network anomaly detection system. In *Journal of Physics: Conference Series*, volume 1018, page 012015. IOP Publishing, 2018.
- [170] Beny Nugraha, Anshitha Nambiar, and Thomas Bauschert. Performance evaluation of botnet detection using deep learning techniques. In *2020 11th International Conference on Network of the Future (NoF)*, pages 141–149. IEEE, 2020.
- [171] Chris Nunnery, Greg Sinclair, and Brent ByungHoon Kang. Tumbling down the rabbit hole: Exploring the idiosyncrasies of botmaster systems in a multi-tier botnet infrastructure. In *LEET*, 2010.
- [172] Dyego HL Oliveira et al. Adaptive model for network resources prediction in modern internet service providers. In *IEEE ISCC*, 2020.
- [173] Eva Ostertagova and Oskar Ostertag. Forecasting using simple exponential smoothing method. *Acta Electrotechnica et Informatica*, 12:62–66, 12 2012. doi:10.2478/v10198-012-0034-2.
- [174] Jiawei Han others. Efficient Mining of Partial Periodic Patterns in Time Series Database. In *15th International Conference on Data Engineering*, pages 106–115. IEEE Computer Society, 1999. doi:10.1109/ICDE.1999.754913.
- [175] Dimou Paraskevi, Jan Fajfer, Nicolas Müller, Eva Papadogiannaki, Evangelos Reklitis, and František Strásák. *Encrypted Traffic Analysis, Use Cases & Security Challenges*. 2020. URL: <https://www.enisa.europa.eu/publications/encrypted-traffic-analysis>.
- [176] Sergio Pastrana et al. A first look at the crypto-mining malware ecosystem: A decade of unrestricted wealth. In *Proceedings of the Internet Measurement Conference, IMC '19*, page 73–86, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3355369.3355576.

- [177] Constantinos Patsakis, Fran Casino, and Vasilios Katos. Encrypted and covert dns queries for botnets: Challenges and countermeasures. *Computers & Security*, 88:101614, 2020.
- [178] Karl Pearson. Notes on the history of correlation. *Biometrika*, 13(1):25–45, 1920. URL: <http://www.jstor.org/stable/2331722>.
- [179] Jian Pei and Yang Lu. Forecasting natural disaster frequencies using nonstationary count time series models. *Statistical Papers*, 66(3):1–44, 2025.
- [180] Richard Plný et al. *Datasets of Cryptomining Communication*. Zenodo, October 2022. doi:10.5281/zenodo.7189293.
- [181] Richard Plný, Karel Hynek, and Tomáš Čejka. DeCrypto: Finding Cryptocurrency Miners on ISP Networks. In *NordSec 2022*. Springer, 2022. doi:10.1007/978-3-031-22295-5_8.
- [182] Jon Postel. TCP Maximum Segment Size and Related Topics. RFC 879, RFC Editor, November 1983. <http://www.rfc-editor.org/rfc/rfc879.txt>. URL: <http://www.rfc-editor.org/rfc/rfc879.txt>.
- [183] Andreas Prokoph and Franz Barthelmes. Detection of nonstationarities in geological time series: wavelet transform of chaotic and cyclic sequences. *Computers & Geosciences*, 22(10):1097–1108, 1996.
- [184] Tom Puech, Matthieu Boussard, Anthony D’Amato, and Gaëtan Millerand. A fully automated periodicity detection in time series. In *International Workshop on Advanced Analysis and Learning on Temporal Data*, pages 43–54. Springer, 2020.
- [185] Muhammad Aidiel Rachman Putra, Tohari Ahmad, and Dandy Pramana Hostiadi. Analysis of botnet attack communication pattern behavior on computer networks. *International Journal of Intelligent Engineering & Systems*, 15(4), 2022.
- [186] Yong Qiao et al. Detecting P2P bots by mining the regional periodicity. *Journal of Zhejiang University SCIENCE C*, 14(9):682–700, Sep 2013. doi:10.1631/jzus.C1300053.
- [187] M Nene Rajkumar. A survey on latest dos attacks: classification and defense mechanisms. *International Journal of Innovative Research in Computer and Communication Engineering*, 1(8):1847–1860, 2013.
- [188] Nipun Ramakrishnan et al. Network traffic prediction using recurrent neural networks. In *IEEE ICMLA*, 2018.
- [189] Suhasini Subba Rao. A course in time series analysis. *Technical Report, Texas A&M University*, 2022.

-
- [190] Faras Rasheed, Mohammed Alshalalfa, and Reda Alhajj. Efficient periodicity mining in time series databases using suffix trees. *IEEE Transactions on Knowledge and Data Engineering*, 23(1):79–94, 2010.
 - [191] Xiaoli Ren, Xiaoyong Li, Kaijun Ren, Junqiang Song, Zichen Xu, Kefeng Deng, and Xiang Wang. Deep learning-based weather prediction: a survey. *Big Data Research*, 23:100178, 2021.
 - [192] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018. URL: <https://www.rfc-editor.org/info/rfc8446>, doi:10.17487/RFC8446.
 - [193] Eric Rescorla et al. TLS Encrypted Client Hello. Internet-Draft draft-ietf-tls-esni-16, Internet Engineering Task Force, 2023.
 - [194] Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in Neural Information Processing Systems*, 32, 2019.
 - [195] Ganesh Sadasivan, Nevil Brownlee, Benoît Claise, and Juergen Quittek. Architecture for IP Flow Information Export. RFC 5470, March 2009. URL: <https://www.rfc-editor.org/info/rfc5470>, doi:10.17487/RFC5470.
 - [196] Amir Mahdi Sadeghzadeh et al. Adversarial Network Traffic: Towards Evaluating the Robustness of Deep-Learning-based Network Traffic Classification. *IEEE Trans. Netw. Serv. Manag.*, 18(2):1962–1976, 2021. doi:10.1109/TNSM.2021.3052888.
 - [197] Mahmood Safaei, Shahla Asadi, Maha Driss, Wadii Boulila, Abdullah Alsaeedi, Hassan Chizari, Rusli Abdullah, and Mitra Safaei. A systematic literature review on outlier detection in wireless sensor networks.
 - [198] Lukasz Saganowski et al. Time Series Forecasting With Model Selection Applied to Anomaly Detection In Network Traffic. *Log. J. IGPL*, 28(4):531–545, 2020. doi:10.1093/jigpal/jzz059.
 - [199] Łukasz Saganowski and Tomasz Andrysiak. Time series forecasting with model selection applied to anomaly detection in network traffic.
 - [200] Amiya Kumar Sahu et al. Internet of things attack detection using hybrid deep learning model. *Computer Communications*, 176:146–154, 2021.
 - [201] Mustafa Al Samara, Ismail Bennis, Abdelhafid Abouaissa, and Pascal Lorenz. A survey of outlier detection techniques in IoT: Review and classification.
 - [202] Debmalaya Sarkar et al. Detection of Tor Traffic using Deep Learning. In *AICCSA 2020*, pages 1–8. IEEE, 2020. doi:10.1109/AICCSA50499.2020.9316533.

- [203] Jeffrey Scargle. Studies in astronomical time series analysis. II - Statistical aspects of spectral analysis of unevenly spaced data. *The Astrophysical Journal*, 263, December 1982. doi:10.1086/160554.
- [204] Jeffrey D Scargle. Studies in astronomical time series analysis. iii-fourier transforms, autocorrelation functions, and cross-correlation functions of unevenly spaced data. *Astrophysical Journal, Part 1 (ISSN 0004-637X)*, vol. 343, Aug. 15, 1989, p. 874-887., 343:874–887, 1989.
- [205] Dominik Schatzmann, Wolfgang Mühlbauer, Thrasyvoulos Spyropoulos, and Xenofontas A. Dimitropoulos. Digging Into HTTPS: Flow-based Classification of Webmail Traffic. In Mark Allman, editor, *Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference, IMC 2010, Melbourne, Australia - November 1-3, 2010*, pages 322–327. ACM, 2010. doi:10.1145/1879141.1879184.
- [206] Eric D. Scheirer et al. Construction and Evaluation of a Robust Multifeature Speech/Music Discriminator. In *ICASSP 1997*, pages 1331–1334. IEEE Computer Society, 1997. doi:10.1109/ICASSP.1997.596192.
- [207] Arthur Schuster et al. *Periodogram Analysis*. 1898.
- [208] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.
- [209] Muhammad Shafiq et al. Selection of Effective Machine Learning Algorithm and Bot-IoT Attacks Traffic Identification for Internet of Things in Smart City. *Future Gener. Comput. Syst.*, 107:433–442, 2020. doi:10.1016/j.future.2020.02.017.
- [210] Shahab Shamsirband and Anthony T Chronopoulos. A new malware detection system using a high performance-elm method. In *Proceedings of the 23rd international database applications & engineering symposium*, pages 1–10, 2019.
- [211] Danil Shamsimukhametov, Anton Kurapov, Mikhail Liubogoshchev, and Evgeny Khorov. Is encrypted clienthello a challenge for traffic classification? *IEEE Access*, 10:77883–77897, 2022.
- [212] Tal Shapira et al. Flowpic: Encrypted internet traffic classification is as easy as image recognition. In *IEEE INFOCOM 2019*. IEEE, 2019.
- [213] Iman Sharafaldin et al. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018.
- [214] Shi Shen, Changxiu Cheng, Changqing Song, Jing Yang, Shanli Yang, Kai Su, Lihua Yuan, and Xiaoqiang Chen. Spatial distribution patterns of global natural disasters based on biclustering. *Natural hazards*, 92(3):1809–1820, 2018.

-
- [215] Wan-Chen Shi et al. Deepbot: a time-based botnet detection with deep learning. *Soft Computing*, 24:16605–16616, 2020.
- [216] Federico Simmross-Wattenberg, Juan Ignacio Asensio-Perez, Pablo Casaseca-de-la Higuera, Marcos Martin-Fernandez, Ioannis A. Dimitriadis, and Carlos Alberola-Lopez. Anomaly detection in network traffic based on statistical inference and alpha-stable modeling. *IEEE Transactions on Dependable and Secure Computing*, 8(4):494–509, 2011. doi:10.1109/TDSC.2011.14.
- [217] Manmeet; Singh Singh, Maninder; Kaur, and Sanmeet. “10 Days DNS Network Traffic from April-May, 2016”, *Mendeley Data*, V2. doi: 10.17632/zh3wnddzxy.2. doi:10.17632/zh3wnddzxy.2.
- [218] Florian Skopik, Max Landauer, and Markus Wurzenberger. Blind spots of security monitoring in enterprise infrastructures: a survey. *IEEE Security & Privacy*, 20(6):18–26, 2022.
- [219] Abir Smiti. A critical overview of outlier detection methods.
- [220] P. Sonali and D. Nagesh Kumar. Review of trend detection methods and their application to detect temperature changes in india. *Journal of Hydrology*, 476:212–227, 2013. URL: <https://www.sciencedirect.com/science/article/pii/S0022169412009341>, doi:10.1016/j.jhydrol.2012.10.034.
- [221] Ziyang Song, Qincheng Lu, He Zhu, David Buckeridge, and Yue Li. Trajgpt: Irregular time-series representation learning of health trajectory. *IEEE Journal of Biomedical and Health Informatics*, 2025.
- [222] Martim Sousa, Ana Maria Tomé, and José Moreira. Long-term forecasting of hourly retail customer flow on intermittent time series with multiple seasonality. *Data Science and Management*, 5(3):137–148, 2022.
- [223] Anna Sperotto, Gregor Schaffrath, Ramin Sadre, Cristian Morariu, Aiko Pras, and Burkhard Stiller. An Overview of IP Flow-based Intrusion Detection. *IEEE Commun. Surv. Tutorials*, 12(3):343–356, 2010. doi:10.1109/SURV.2010.032210.00054.
- [224] Cédric St-Onge, Nadja Kara, Omar Abdel Wahab, Claes Edstrom, and Yves Lemieux. Detection of time series patterns and periodicity of cloud computing workloads. *Future Generation Computer Systems*, 109:249–261, 2020.
- [225] Stuart Staniford, James A Hoagland, and Joseph M McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10(1-2):105–136, 2002.
- [226] George Stergiopoulos et al. Automatic Detection of Various Malicious Traffic Using Side Channel Features on TCP Packets. In *ESORICS 2018*, volume 11098, pages 346–362. Springer, 2018. doi:10.1007/978-3-319-99073-6_17.

- [227] James H Stock and Mark W Watson. Twenty years of time series econometrics in ten pictures. *Journal of Economic Perspectives*, 31(2):59–86, 2017.
- [228] Petre Stoica, Randolph L Moses, et al. *Spectral analysis of signals*, volume 452. Pearson Prentice Hall Upper Saddle River, NJ, 2005.
- [229] A. Suarez et al. Analytical Comparison Between Time- And Frequency-domain Techniques for Phase-noise Analysis. *IEEE Transactions on Microwave Theory and Techniques*, 50(10):2353–2361, 2002. doi:10.1109/TMTT.2002.803457.
- [230] Chenxi Sun, Shenda Hong, Moxian Song, and Hongyan Li. A review of deep learning methods for irregularly sampled medical time series data. *arXiv preprint arXiv:2010.12493*, 2020.
- [231] Gian Antonio Susto, Angelo Cenedese, and Matteo Terzi. Chapter 9 - time-series classification methods: Review and applications to power systems data. In Reza Arghandeh and Yuxun Zhou, editors, *Big Data Application in Power Systems*, pages 179–220. Elsevier, 2018. URL: <https://www.sciencedirect.com/science/article/pii/B9780128119686000097>, doi:10.1016/B978-0-12-811968-6.00009-7.
- [232] Thomas L. Szabo. 5 - Transducers. In Thomas L. Szabo, editor, *Diagnostic Ultrasound Imaging*, Biomedical Engineering, pages 97–135. Academic Press, Burlington, 2004. doi:10.1016/B978-012680145-3/50006-2.
- [233] Rui Tanabe, Tatsuya Tamai, Akira Fujita, Ryoichi Isawa, Katsunari Yoshioka, Tsutomu Matsumoto, Carlos Gañán, and Michel Van Eeten. Disposable botnets: examining the anatomy of iot botnet infrastructure. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, pages 1–10, 2020.
- [234] Aleem Dad Khan Tareen, Malik Sajjad Ahmed Nadeem, Kimberlee Jane Kearfott, Kamran Abbas, Muhammad Asim Khawaja, and Muhammad Rafique. Descriptive analysis and earthquake prediction using boxplot interpretation of soil radon time series data. *Applied Radiation and Isotopes*, 154:108861, 2019. URL: <https://www.sciencedirect.com/science/article/pii/S0969804319307353>, doi:10.1016/j.apradiso.2019.108861.
- [235] Imad Tareq et al. Analysis of ton-iot, unw-nb15, and edge-iiot datasets using dl in cybersecurity for iot. *Applied Sciences*, 12(19):9572, 2022.
- [236] Farhan Tariq and Shamim Baig. Botnet Classification Using Centralized Collection of Network Flow Counters in Software Defined Networks. *International Journal of Computer Science and Information Security*, 14(8):1075, 2016.
- [237] Rosemary Tawn, Jethro Browell, and Iain Dinwoodie. Missing data in wind farm time series: Properties and effect on forecasts. *Electric Power Systems Research*, 189:106640, 2020.

-
- [238] Simon Nam Thanh Vu, Mads Stege, Peter Issam El-Habr, Jesper Bang, and Nicola Dragoni. A survey on botnets: Incentives, evolution, detection and current trends. *Future Internet*, 13(8):198, 2021.
- [239] Denis Tikunov et al. Traffic prediction for mobile network using holt-winter’s exponential smoothing. In *IEEE SoftCOM*, 2007.
- [240] Yuerong Tong, Jingyi Liu, Lina Yu, Liping Zhang, Linjun Sun, Weijun Li, Xin Ning, Jian Xu, Hong Qin, and Qiang Cai. Technology investigation on time series classification and prediction. *PeerJ Computer Science*, 8:e982, 2022.
- [241] Zdena Tropková et al. Novel HTTPS Classifier Driven by Packet Bursts, Flows, and Machine Learning. In *CNSM 2021*. IEEE, 2021. doi:10.23919/CNSM52442.2021.9615561.
- [242] Zisis Tsiatsikas, Georgios Karopoulos, and Georgios Kambourakis. Measuring the adoption of tls encrypted client hello extension and its forebear in the wild. In *European Symposium on Research in Computer Security*, pages 177–190. Springer, 2022.
- [243] Daniel Uhříček, Karel Hynek, Tomáš Čejka, and Dušan Kolář. Bota: Explainable iot malware detection in large networks. *IEEE Internet of Things Journal*, 10(10):8416–8431, 2023. doi:10.1109/JIOT.2022.3228816.
- [244] Usman Ahmad Usmani, Ari Happonen, and Junzo Watada. A review of unsupervised machine learning frameworks for anomaly detection in industrial applications. In *Intelligent Computing*.
- [245] Jacob T. VanderPlas. Understanding the lomb–scargle Periodogram. *The Astrophysical Journal Supplement Series*, 236(1):16, may 2018. doi:10.3847/1538-4365/aab766.
- [246] Dmitrii Vekshin et al. DoH Insight: Detecting DNS Over HTTPS by Machine Learning. In *ARES 2020*. ACM, 2020. doi:10.1145/3407023.3409192.
- [247] Petr Velan, Milan Čermák, Pavel Čeleda, and Martin Drašar. A survey of methods for encrypted traffic classification and analysis. *International Journal of Network Management*, 25(5):355–374, 2015.
- [248] D Vere-Jones and RB Davies. A statistical survey of earthquakes in the main seismic region of new zealand: Part 2—time series analyses. *New Zealand journal of geology and geophysics*, 9(3):251–284, 1966.
- [249] Aditya Vikram and Mohana. Anomaly detection in network traffic using unsupervised machine learning approach. In *ICCES*.

- [250] Roberto Vio, María Diaz-Trigo, and Paola Andreani. Irregular time series in astronomy and the use of the lomb–scargle periodogram. *Astronomy and Computing*, 1:5–16, 2013.
- [251] Ly Vu et al. Time Series Analysis for Encrypted Traffic Classification: A Deep Learning Approach. In *ISCIT 2018*. IEEE, 2018.
- [252] Lu Wang, Chaoyun Zhang, Ruomeng Ding, Yong Xu, Qihang Chen, Wentao Zou, Qingjun Chen, Meng Zhang, Xuedong Gao, Hao Fan, et al. Root cause analysis for microservice systems via hierarchical reinforcement learning from human feedback. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5116–5125, 2023.
- [253] Wei Wang et al. End-to-end Encrypted Traffic Classification With One-dimensional Convolution Neural Networks. In *ISI 2017*, pages 43–48. IEEE, 2017. doi:10.1109/ISI.2017.8004872.
- [254] Wei Wang et al. Malware Traffic Classification Using Convolutional Neural Network For Representation Learning. In *ICOIN 2017*, pages 712–717. IEEE, 2017. doi:10.1109/ICOIN.2017.7899588.
- [255] Chongbo Wei, Gaogang Xie, and Zulong Diao. A lightweight deep learning framework for botnet detecting at the iot edge. *Computers & Security*, 129:103195, 2023.
- [256] William Wei. *Time Series Analysis: Univariate and Multivariate Methods*, volume 33. 01 1989. doi:10.2307/2289741.
- [257] Qingsong Wen et al. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022.
- [258] Qingsong Wen, Kai He, Liang Sun, Yingying Zhang, Min Ke, and Huan Xu. Robustperiod: Robust time-frequency mining for multiple periodicity detection. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2328–2337, 2021.
- [259] Gerald Woo et al. Etsformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381*, 2022.
- [260] S. J Worley et al. Comparison of Time Domain and Frequency Domain Variables From the Signal-averaged Electrocardiogram: A Multivariable Analysis. *Journal of the American College of Cardiology*, 1988. doi:10.1016/S0735-1097(98)90064-X.
- [261] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

-
- [262] Renjie Wu and Eamonn J Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE transactions on knowledge and data engineering*, 35(3):2421–2429, 2021.
- [263] Shin-Fu Wu, Chia-Yung Chang, and Shie-Jue Lee. Time series forecasting with missing values. In *2015 1st International Conference on Industrial Networks and Intelligent Systems (INISCom)*, pages 151–156. IEEE, 2015.
- [264] Asrul H Yaacob, Ian KT Tan, Su Fong Chien, and Hon Khi Tan. Arima based network anomaly detection. In *2010 Second International Conference on Communication Software and Networks*, pages 205–209. IEEE, 2010.
- [265] Ruoyu Yan and Ran Liu. Principal component analysis based network traffic classification. *Journal of Computers*, 9(5):1234–1240, 2014. doi:10.4304/jcp.9.5.1234-1240.
- [266] Yang Yang et al. A network traffic classification method based on dual-mode feature extraction and hybrid neural networks. *IEEE Transactions on Network and Service Management*, 2023.
- [267] Sungwoong Yeom, Chulwoong Choi, and Kyungbaek Kim. Autoencoder based feature extraction for multi-malicious traffic classification. SMA 2020, page 285–287, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3426020.3426093.
- [268] Tao Yi, Xingshu Chen, Yi Zhu, Weijing Ge, and Zhenhui Han. Review on the application of deep learning in network attack detection. *Journal of Network and Computer Applications*, 2023.
- [269] Hao Yin et al. Network Traffic Prediction Based on a New Time Series Model. *Int. J. Commun. Syst.*, 18(8):711–729, 2005. doi:10.1002/dac.721.
- [270] P Young and Stephen Shellswell. Time series analysis, forecasting and control. *IEEE Transactions on Automatic Control*, 17(2):281–283, 1972.
- [271] Tahmina Zebin et al. An explainable ai-based intrusion detection system for dns over https (doh) attacks. *IEEE Transactions on Information Forensics and Security*, 17:2339–2349, 2022. doi:10.1109/TIFS.2022.3183390.
- [272] Scott L. Zeger, Rafael Irizarry, and Roger D. Peng. On time series analysis of public health and biomedical data. *Annual Review of Public Health*, 27(1):57–79, 2006. PMID: 16533109. arXiv:https://doi.org/10.1146/annurev.publhealth.26.021304.144517, doi:10.1146/annurev.publhealth.26.021304.144517.
- [273] Ailing Zeng et al. Are transformers effective for time series forecasting? In *AAAI*, 2023.

- [274] Zili Zha, An Wang, Yang Guo, Doug Montgomery, and Songqing Chen. Botsifter: an sdn-based online bot detection framework in data centers. In *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 142–150. IEEE, 2019.
- [275] Yaning Zhang, Weishi Sun, and Shuzhuang Zhang. Identify vpn traffic under https tunnel using three-dimensional sequence features. In *Proceedings of the 2022 11th International Conference on Networks, Communication and Computing*, pages 18–23, 2022.
- [276] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.
- [277] Indrė Žliobaitė, Mykola Pechenizkiy, and Joao Gama. An overview of concept drift applications. *Big data analysis: new algorithms for a new society*, pages 91–114, 2016.

Publications of the Thesis Author

The number of citations (excluding autocitations) written below each paper was obtained on January 5, 2026, from two platforms, Google Scholar and ResearchGate. Overall, the publications of the thesis author received 82 citations (excluding autocitations). Moreover, the published datasets are listed below, along with the number of individuals who have downloaded each dataset (as reported by the Zenodo platform on January 5, 2026). Overall, 21,774 individuals downloaded the datasets created by the thesis author.

Reviewed Publications Relevant to the Thesis

- [A.1] **Koumar, J.**, Čejka, T. *Unevenly spaced time series from network traffic*. In *2023 7th Network Traffic Measurement and Analysis Conference (TMA)*, pages 1–4, IEEE, 2023.

The paper has been cited in 1 paper.

- [A.2] **Koumar, J.**, Hynek, K., Čejka, T. *Network traffic classification based on single flow time series analysis*. In *2023 19th International Conference on Network and Service Management (CNSM)*, pages 1–7, IEEE, 2023.

The paper has been cited in 29 papers.

- [A.3] **Koumar, J.**, Hynek, K., Pešek, J., Čejka, T. *Nettisa: Extended ip flow with time-series features for universal bandwidth-constrained high-speed network traffic classification*. *Computer Networks*, Volume 240, pages 110147, Elsevier, 2024.

The paper has been cited in 18 papers.

- [A.4] **Koumar, J.**, Čejka, T. *Network traffic classification based on periodic behavior detection*. In *2022 18th International Conference on Network and Service Management (CNSM)*, pages 359–363, IEEE, 2022.

The paper has been cited in 13 papers.

- [A.5] **Koumar, J.**, Plný, R., Čejka, T. *Enhancing DeCrypto: Finding Cryptocurrency Miners Based on Periodic Behavior*. In *2023 19th International Conference on Network and Service Management (CNSM)*, pages 1–7, IEEE, 2023.

The paper has been cited in 0 papers.

- [A.6] Oškera, D., **Koumar, J.**, Pokorná, A., Jeřábek, K., and Čejka, T. *Botnet Detection Through Periodic Patterns in Command-and-Control Network Traffic*. In *2025 21th International Conference on Network and Service Management (CNSM)*, pages 1–6, IEEE, 2025.

The paper has been cited in 0 papers.

- [A.7] **Koumar, J.**, Hynek, K., Čejka, T., Šiška, P. *CESNET-TimeSeries24: Time Series Dataset for Network Traffic Anomaly Detection and Forecasting*. Scientific Data, Volume 12, Number 1, pages 338, Nature Publishing Group UK London, 2025.

The paper has been cited in 13 papers.

- [A.8] **Koumar, J.**, Pešek, J., Jeřábek, K., Čejka, T. *Towards building network outlier detection system for network traffic monitoring*. In *NOMS 2025-2025 IEEE Network Operations and Management Symposium*, pages 1–6, IEEE, 2025.

The paper has been cited in 1 paper.

- [A.9] Jeřábek, K., **Koumar, J.**, Setinský, J., Pešek, J. *Explainable Anomaly Detection in Network Traffic Using LLM*. In *NOMS 2025-2025 IEEE Network Operations and Management Symposium*, pages 1–6, IEEE, 2025.

The paper has been cited in 2 papers.

Remaining Reviewed Publications

- [A.10] Pešek, J., Plný, R., **Koumar, J.**, Jeřábek, K., Čejka, T. *Augmenting monitoring infrastructure for dynamic software-defined networks*. In *2023 8th International Conference on Smart and Sustainable Technologies (SpliTech)*, pages 1–4, IEEE, 2023.

The paper has been cited in 2 papers.

- [A.11] Jančíčka, L., **Koumar, J.**, Soukup, D., Čejka, T. *Analysis of statistical distribution changes of input features in network traffic classification domain*. In *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, pages 1–4, IEEE, 2024.

The paper has been cited in 1 paper.

- [A.12] Jančíčka, L., Soukup, D., **Koumar, J.**, Němec, F., Čejka, T. *MFWD: Model-based Feature Weight Drift Detection Showcased on TLS and QUIC Traffic*. In *2024 20th International Conference on Network and Service Management (CNSM)*, pages 1–5, IEEE, 2024.

The paper has been cited in 1 paper.

- [A.13] Kureš, M., **Koumar, J.**, and Hynek, K. *CESNET TS-Zoo: A Library for Reproducible Analysis of Network Traffic Time Series*. In *2025 21th International Conference on Network and Service Management (CNSM)*, pages 1–5, IEEE, 2025.

The paper has been cited in 0 papers.

- [A.14] Hulák, M., Bartoš, V., **Koumar, J.**, and Čejka, T. *DAF: Device Annotation Framework*. In *2025 21th International Conference on Network and Service Management (CNSM)*, pages 1–6, IEEE, 2025.

The paper has been cited in 0 papers.

- [A.15] **Koumar, J.**, Smoleň, T., Jeřábek, K., Čejka, T. *Comparative Analysis of Deep Learning Models for Real-World ISP Network Traffic Forecasting*. in *IEEE Transactions on Network and Service Management* doi: 10.1109/TNSM.2025.3636557 2025.

The paper has been cited in 1 paper.

Remaining Publications

- [A.16] Jeřábek, K., Luxemburk, J., Plný, R., **Koumar, J.**, Pešek, J., Hynek, K. *When Simple Model Just Works: Is Network Traffic Classification in Crisis?*. arXiv preprint arXiv:2506.08655, 2025. Submitted to the Euro S&P 2026 conference.

- [A.17] Mudruňka, K., **Koumar, J.**, Jeřábek, K. *Device Type Classification on ISP Network using Time Series Analysis*. Submitted to the NOMS 2026 conference.

Published Datasets

- [A.18] **Koumar, J.**, Čejka, T. (2023). “CESNET-USTS23: a benchmark dataset of Unevenly spaced time series from network traffic” [Data set]. Zenodo. <https://doi.org/10.5281/zenodo.7923745>

The dataset was downloaded by 699 individuals.

- [A.19] **Koumar, J.**, Plný, R., Čejka, T. (2023). “CESNET-MINER22-TS: Periodic Behavior Features of Cryptomining Communication” [Data set]. Zenodo. <https://doi.org/10.5281/zenodo.8033351>

The dataset was downloaded by 246 individuals.

- [A.20] **Koumar, J.**, Hynek, K., Čejka, T. (2023). “Network traffic datasets created by Single Flow Time Series Analysis” [Data set]. Zenodo. <https://doi.org/10.5281/zenodo.8035724>

The dataset was downloaded by 6,679 individuals.

- [A.21] **Koumar, J.**, Hynek, K., Pešek, J., Čejka, T. (2023). “Network traffic datasets with novel extended IP flow called NetTiSA flow” [Data set]. Zenodo. <https://doi.org/10.5281/zenodo.8301043>

The dataset was downloaded by 3,506 individuals.

- [A.22] **Koumar, J.**, Hynek, K., Čejka, T., Šiška, P. (2024). CESNET-TimeSeries24: Time Series Dataset for Network Traffic Anomaly Detection and Forecasting [Data set]. Zenodo. <https://doi.org/10.5281/zenodo.13382427>

The dataset was downloaded by 10,561 individuals.

- [A.23] Oškera, D., **Koumar, J.**, Pokorná, A., Jeřábek, K., Čejka, T. (2025). CESNET-CC25: Long-term capture of C&C communication of botnets in the PCAP format [Data set]. Zenodo. <https://doi.org/10.5281/zenodo.16752462>

The dataset was downloaded by 8 individuals.

- [A.24] Oškera, D., **Koumar, J.**, Pokorná, A., Jeřábek, K., Čejka, T. (2025). CESNET-CC25: Dataset for detection of C&C communication of botnets in the IP flow format [Data set]. Zenodo. <https://doi.org/10.5281/zenodo.16753890>

The dataset was downloaded by 43 individuals.

- [A.25] Oškera, D., **Koumar, J.**, Pokorná, A., Jeřábek, K., Čejka, T. (2025). CESNET-CC25: Dataset for detection of C&C communication of botnets in the periodic behavior features from multiframe time series format [Data set]. Zenodo. <https://doi.org/10.5281/zenodo.16753981>

The dataset was downloaded by 26 individuals.

- [A.26] Mudruňka, K., **Koumar, J.**, Jeřábek, K. (2025). CESNET-DeviceType24: Dataset for Device Type Classification on ISP Network [Data set]. Zenodo. <https://doi.org/10.5281/zenodo.17542827>

The dataset was downloaded by 14 individuals.