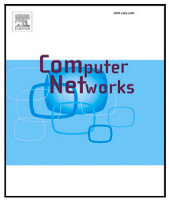
Computer Networks 240 (2024) 110147

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



NetTiSA: Extended IP flow with time-series features for universal bandwidth-constrained high-speed network traffic classification

Josef Koumar [a,*], Karel Hynek [b], Jaroslav Pešek [a], Tomáš Čejka [b]

[a] Czech Technical University in Prague, Faculty of Information Technology, Thákurova 9, 160 00 Prague 6, Czech Republic
[b] CESNET a.l.e., Generála Píky 430/26, 160 00 Prague 6, Czech Republic

ARTICLE INFO

Keywords:
Time series
Unevenly spaced time series
Time series analysis
Classification
Computer networks
Machine learning
IP flow
Flow exporter

ABSTRACT

Network traffic monitoring based on IP Flows is a standard monitoring approach that can be deployed to various network infrastructures, even the large ISP networks connecting millions of people. Since flow records traditionally contain only limited information (addresses, transport ports, and amount of exchanged data), they are also commonly extended by additional features that enable network traffic analysis with high accuracy. These flow extensions are, however, often too large or hard to compute, which then allows only offline analysis or limits their deployment only to smaller-sized networks. This paper proposes a novel extended IP flow called NetTiSA (Network Time Series Analysed) flow, based on analysing the time series of packet sizes. By thoroughly testing 25 different network traffic classification tasks, we show the broad applicability and high usability of NetTiSA flow. For practical deployment, we also consider the sizes of flows extended by NetTiSA features and evaluate the performance impacts of their computation in the flow exporter. The novel features proved to be computationally inexpensive and showed excellent discriminatory performance. The trained machine learning classifiers with proposed features mostly outperformed the state-of-the-art methods. NetTiSA finally bridges the gap and brings universal, small-sized, and computationally inexpensive features for traffic classification that can be scaled up to extensive monitoring infrastructures, bringing the machine learning traffic classification even to 100 Gbps backbone lines.

1. Introduction

Network monitoring plays a crucial role in the overall computer security management. Compared to protections (such as AntiVirus software) deployed on the end devices, the network-based intrusion detection and prevention systems can protect infrastructure against users' sloppiness, policy violations, or (at worst) intentional attacks from the inside. Moreover, network security plays an essential role in end-device protection, where anti-virus software availability is limited or even non-existent, such as IoT devices [1] or sensors [2] that are commonly infected by Distributed Denial of Service (DDoS) botnets [3]. However, maintaining network security has become increasingly challenging in recent years due to mass traffic encryption and consequent reduced visibility. The encryption of TLS certificates by TLS1.3 [4], deployment of encrypted DNS [5], or Encrypted Client Hello proposal [6] removed the few-remaining information essential for effective threat detection. Therefore, if possible, the security managers are forced to deploy intermediate proxies to decrypt the traffic [7] and inspect it via Deep Packet Inspection (DPI) tools such as Suricata.[1] In such cases, the deployment of the intermediate proxy is much more intrusive than sending domain names and certificates in plaintext. The DPI 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 [8].

Internet Service Providers (ISP) need to perform network monitoring and cybersecurity threat detection to force internal policies, infrastructure protection, and lines overloading prevention to maintain service [8]. The single intrusion detector deployed at the ISP level can protect many users (even in order of millions) and prevent or minimise the impact of attacks, such as 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 100 Gbps backbone lines with DPI. Therefore, large-scale infrastructures are often monitored using the flow-based approach, representing the communication between two devices in the form of flows. The flows are created at observation points—network devices aggregating raw network traffic

* Corresponding author.
E-mail address: koumajos@fit.cvut.cz (J. Koumar).

[1] https://suricata.io

https://doi.org/10.1016/j.comnet.2023.110147
Received 9 October 2023; Received in revised form 1 December 2023; Accepted 17 December 2023
Available online 3 January 2024

into flow records. The flow records are then transmitted to the flow collector using a flow-export protocol such as the *Internet Protocol Flow Information Export (IPFIX) [9]* or NetFlow Version 9 [10].

Even though there are not any rigid specifications of flow, traditionally, it contains only IP addresses, transport ports, and the amount of transferred data [11]. Raw flows have been previously successfully used in volumetric attack detection, such as DDoS attacks [12]. Detection of traffic types which cannot be distinguished volumetrically is challenging with traditional flows and often requires additional data source or active probing confirmation [13], which increases the load on the already resource-constrained monitoring hardware. To avoid active probing or information gathering from external sources, researchers perform various flow extensions to provide additional information that enables accurate classification of network traffic [14–16] or even threat detection [17,18]. Moreover, with the rise of traffic encryption, the flows are often extended by non-payload-based data about the connections, which also allows traffic classification with encrypted payloads. This extended information often contains statistics about transferred packets and inter-packet times.

There are many proposals of detection and classification methods in the field of network security that use extended flows combined with multiple techniques, including machine learning. However, these works mainly focus on model accuracy and not practical usability; thus, we can see almost no real-world deployment of these approaches. The feasibility of deployment is usually not a concern. Even though there are many proposals [19–21] for complex ML offloading to scale up the deployment and improve efficiency, many traffic detectors require traffic features that are too resource-intensive for computation in real-time. Their computation and memory requirements then allow their deployment only in an offline manner, as discussed by Jerabek et al. [13].

Other approaches extend the flows for too much data, such as a sequence of packet lengths and times [22], sequence of packet bursts and times [23], initial-data-packet content [24], or simply too many individual features that increase the flow size several times as in the case of CICFlowMeter.[2] The size of the flow telemetry matters for deployment since it occupies bandwidth that could be otherwise commercialized, and it significantly increases the performance requirements on the monitoring infrastructure that needs to process more data in a given time. On the contrary, the detectors with small feature vectors (such as [25]) are not universal; their specifically tailored features were proved to work on a single task, but other tasks were not evaluated. Thus, there is still a need for a small but universal feature vector, easily extractable from network traffic, to enable encrypted network traffic classification at scale.

Since there are no proposals for such universal feature vectors, we explored the possibility of their creation. We build this research on top of our previous study [26], which uses *Time Series Analysis (TSA)* of *Single Flow Time Series (SFTS)* [27]. The TSA approach can capture detailed information from both long-lasting and short flows and applies to multiple network classification tasks. In TSA, each flow is internally represented as a time series of network packets, i.e., one datapoint of SFTS describes a packet payload size with its position in time series defined by the packet's transmission time. Nevertheless, the approach proposed in [26] is usable only for offline analysis due to high computational complexity and memory requirements of TSA.

Compared to the previous study, we aimed to create a universal set of features that can be easily extracted from high-speed network lines, i.e., the extraction process must have low memory and computational complexity. The proposed feature set can be computed online without the necessity of saving the packet time series into the operational memory. The optimized TSA features are called NetTiSA (Network Time Series Analysed) and can be computed directly at the observation point (flow exporter) and exported inside an extended (NetTiSA) flow. The novel feature set describes time dependencies between packets, packet length sequences, and distribution of packet lengths.

To validate the usability of the NetTiSA features, we evaluated them using well-known network traffic monitoring tasks, published datasets, and machine learning (ML) methods. According to our validation, the proposed NetTiSA features are usable in both binary and multiclass network traffic classification problems. Moreover, our approach outperforms a majority of best-performing related works of classification by IP flows on the same datasets while requiring minimal bandwidth for the telemetry compared to related works.

The contributions of our work can be summarized as follows:

- We propose a novel universal approach that uses time series analysis inside the IP flow exporter to generate 13 features included in the extended IP flow called NetTiSA flow. Furthermore, seven more features are computed from the NetTiSA flow and traditional bidirectional flow features before classification, resulting in a universal and compact feature set with 20 features called Enhanced NetTiSA.
- We evaluate the usability of novel NetTiSA flow in multiple binary network traffic classification tasks. ML models trained on Enhanced NetTiSA achieved high Accuracy and F1 scores, mostly outperforming the state-of-the-art classifiers from related works. The evaluated tasks include a detection of Botnet, Cryptomining, DoH, (D)DoS, Malicious DNS, Intrusion, IoT Malware, Tor, and VPN.
- We also evaluate the usability of novel NetTiSA flow in multiclass classification. ML models trained on Enhanced NetTiSA achieved high Accuracy and F1 scores that exceeded the best previous results from relevant works using the same datasets. The multiclass classification tasks include Botnet, IDS, IoT Malware, Tor, and VPN classification into multiple categories.
- The NetTiSA flow achieves better results with smaller network telemetry compared to related works, is universally applicable to multiple classification problems, and enables high-speed network traffic classification.
- An implementation of the NetTiSA flow is publicly available as a plugin for the open source IP flow exporter *ipfixprobe*.[3] The implementation is suitable for high-speed ISP networks and can process 100 Gbps of network traffic. Indeed, this implementation is currently used to export NetTiSA flows from multiple 100 Gbps lines within the CESNET2[4] network.
- To facilitate reproducibility, we publish datasets containing the Enhanced NetTiSA features created from 15 well-known publicly available datasets. The created datasets are available on Zenodo [28].

This paper is divided as follows: Section 2 summarizes the related work on the flow-based network traffic classification. Section 3 describes the selected datasets and network traffic classification use cases. Section 4 provides information on time series analysis concepts and motivation and describes the novel approach to time series analysis in the IP flow exporter. Section 5 provides a complete description of the features exported in the novel extended IP flow. Section 6 describes the complete classification pipeline and presents the results of using the NetTiSA flow for classification. Section 7 analyses the size of the proposed feature vector and compares it with related works. Section 8 provides information about the impact of NetTiSA features calculation, presents a high-speed C++ implementation and compares the performance with other flow extensions. Section 9 concludes this paper.

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

[3] https://github.com/CESNET/ipfixprobe

[4] The Czech Educational and Science Network.

2. Related works

Flow-based network monitoring is essential for telemetry acquisition and security maintenance in large network infrastructures. The flow-based intrusion detection systems 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 flow record has a relatively loose definition [11], 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 so that the users can define their data structures transferred inside flow records. Nowadays, sending variable length lists of common datatypes (such as uint32) or variable length byte arrays is also possible. Thus, it is no wonder that the flows are often extended for various information helpful in intrusion detection. We can divide the flow extensions into three directions: 1. Extension for extracted unencrypted information, 2. Extension for raw packet information, and 3. Extension for precomputed features. These approaches are further described in the following sections.

2.1. Extension for extracted unencrypted information

Exporting unencrypted information from the packet payload is one of the essential functionalities 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 [29]. However, each detection and classification task requires extracting 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, we have seen a trend in traffic encryption and 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) [5,30,31]. Privacy-sensitive information in TLS handshakes is already sometimes encrypted with the Encrypted Client Hello extension [32], which is challenging for traffic classification [33]. Since most of the traffic nowadays is encrypted, the importance of unencrypted information extraction is rapidly decreasing, and it is being replaced by other types of flow extensions targeting encrypted traffic analysis.

2.2. Extension for raw packet information

The extension of flows by packet sequences embeds the raw packet-level information about ongoing connections into the flows. Typically, flows are extended by a *Sequence of Packet Lengths and Times (SPLT)* that aims to export the first n packets of a flow as a sequence of packet lengths (or payload lengths), directions, and inter-packet times [16,34,35]. Each network classifier can then process the SPLT differently to maximize its accuracy. For example, network classifiers proposed by Vekshin et al. [36] or Hynek et al. [37] 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. [16] used SPLT of a length of 30 packets for fine-grained network classification with many labels. They use information from SPLT (packet lengths, directions, and times) as three separate sequences and process them with a Convolutional neural network (CNN) with 1D convolutions. The proposed approach achieved great results in classification since the convolutions used in deep learning can extract discriminatory features themselves [38]. Moreover, they also showed that tree-based ML algorithms can perform with similar accuracy even if raw SPLT is used as an input.

Another use of SPLT was proposed by Chen et al. [39] and also by Shapira et al. [40]. Both approaches embed flows with SPLT sequence into a 2D image, which is then used by CNN with 2D convolutions. The biggest difference between the two approaches is in the size of the used input images. Chen et al. [39] 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. [40] deals with VPN detection tasks and uses SPLT of very long size. Their proposed detector achieved a 99.7% of accuracy; however, it requires $1400x1400$ images as the input, which means that the size of each flow with SPLT sequence is at least 7.48 MB. 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. For example, the flow exporter ipfixprobe limits the size of the SPLT sequence to 30 packets; Cisco Joy[5] software can export SPLT sequences of up to 200 packets—with a default value of 50 packets. The short length then limits the SPLT applicability only to short connections. Therefore, researchers proposed to extend the flows for additional raw features to capture information about the packets that do not fit into the sequence. For example, Tropkova et al. [23] 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 also its length limit, and the aggregation of packets into the bursts loses some information about the exact timing of packets within the burst.

Another approach is to extend flows by a traffic histogram, which was used by Hofstede et al. [41] in the HTTPS brute force detection. The histogram contains information about packets from the whole connection regardless of its length. However, similarly as in SBLT, histograms do not carry 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, Luxembruk et al. [22] used SPLT and also traffic histograms in their QUIC classification task. Aceto et al. [34] and Wang et al. [42] combine SPLT even with unencrypted information from packet headers. The combination with bytes of payload 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.

2.3. Extension for precomputed features

The extensions for raw-packet information, such as SPLT, allow flexibility in feature extraction and allow each detector to compute its own specific discriminatory features; nevertheless, the flows with raw-packet information tend to be larger since 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 exporter is the CICFlowMeter 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 [14,15,43–46].

Similarly, as CICFlowMeter, MontazeriShatoori et al. [47] created a DoHLyzer exporter[6] that produces features directly within the flows, specifically for DoH detection task. These features are, however, entirely different from the CICFlowMeter. The inflexibility of the extensions for precomputed features then forces detectors to use non-optimal

[5] https://github.com/cisco/joy
[6] https://github.com/ahlashkari/DoHLyzer

features (exported either by DoHLyzer or CICFlowMeter), drastically degrading their performance.

The natural solution would be to export as many features as possible to cover a wide variety of 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. [48] 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 used to target novel protocols and encrypted traffic analysis; thus, even a set of 249 features is incomplete.

There are possible approaches for decreasing the size of large feature vectors, such as 1. compressive traffic analysis proposed by Nasr et al. [49], 2. use of embeddings created by Neural Networks instead of classical feature computation as proposed by Sungwoong et al. [50], 3. or feature dimension reduction techniques such as PCA (Principal Component Analysis) [51]. However, we are not aware of any flow exporter that would deploy these techniques to reduce the feature set size and, thus the flow telemetry size. Moreover, all these approaches perform lossy compression that cannot be fully reconstructed back. The feature is then used in its compressed form, which is not understandable by people. Since people often analyse and inspect flows or alerts from the detectors, the lack of understandability then poses significant limitations in real-world deployment [52].

2.4. Merging raw and precomputed features together

Since the use of each flow extension type comes with disadvantages, we explored a hybrid approach between raw and precomputed features that would keep the telemetry size low while still being universal. In our previous work [26], we explored the universality of features created by time-series analysis. The time-series analysis showed excellent universality across multiple use cases; however, its practical deployment was limited due to high feature computational complexity and memory requirements. The novel NetTiSA flow overcomes the computational complexity of time series analysis by thorough optimization, and it finally makes the universal and size-constrained flow extension deployable to production monitoring infrastructure and even scalable to large ISP networks.

3. Datasets

Since we aimed to find a universal feature vector for network traffic classification using machine learning, we needed to select multiple classification tasks that will be used for evaluation. We decided to select tasks based on evaluation data availability—the most studied and important network classification tasks have standard "benchmark" datasets that also allow comparison with the best-performing state-of-the-art classifiers.

We selected 15 different datasets that are provided in the form of raw packet captures (pcaps) and cover the most important detection and classification (even multiclass) tasks in network security. The used datasets for both binary and multiclass classification are written in Table 1 along with the best-performing method. The table also contains the accuracies of the best-performing classifiers as well as their flow extension size.

To find the best-performing classifiers for each dataset, we used common scientific manuscript aggregators such as IEEE Explore,[7] ACM Digital Library,[8] Scopus,[9] and Google Scholar.[10] We either listed articles that referenced concerned datasets or searched for the names of datasets. We went through more than 300 papers and selected the best-performing proposals that met the following conditions, ensuring fair performance comparability with related works: 1. it was a flow-based method; 2. it used the dataset as a whole and classifies all the dataset classes and types of samples; 3. it did not use IP addresses and transport layer ports as input features. The reason behind omitting IP addresses and transport layer ports stems from mainly lab-created datasets usage. Since the variety of IP addresses in the datasets is low, the ML-based detectors tend to overfit these features and overestimate their accuracy; therefore, it is not considered a good practice [60] in this case. Finally, 4. it did not combine the concerned dataset with additional data. The selected methods are then used for performance and telemetry-size comparison with the novel feature vector.

4. Time series analysis

The time series is a natural representation of network packets transferred via the computer networks. The SPLT is an example of time series, where each packet and its timestamp represent a single point. Nevertheless, SPLTs are usually very short, containing about tens of data points. The ipfixprobe flow exporter exports SPLT of length 30, which is sometimes insufficient. Even though the majority of flows on the internet are short—so-called dragonfly flows [79], these statistics are often highly influenced by DNS flows with only two packets. According to Luxemburk et al. [16], more than 15% of TLS flows with successful TLS handshake on the internet are longer than 30 packets, and in the case of QUIC flows, more than 40% of flows are longer than 30 packets [80]. The SPLT sequence then loses information for a non-negligible portion of traffic that could be otherwise used for more accurate detection.

The information loss is also demonstrated in Fig. 1, which shows the percentage of flows that contain more than 30 packets in each selected detection task in Section 3. It can be seen that flow length distribution varies greatly among the classification tasks. In the case of TOR, Cryptomining, and 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.

4.1. Background in time series analysis

The state of the art in time series analysis mostly considers only evenly-spaced times between observations. This type of time series is called evenly spaced (or regularly sampled) [81], and it is defined as the sequence of observations $\{X_n\} = \{x_1, \dots, x_n\}$ taken at times $\{T_n\} = \{t_1, \dots, t_n\}$, where n is the number of observations. It is always true that $t_{j+1} - t_j = t_j - t_{j-1}, \forall j \in \{2, \dots, n-1\}$. Because of this behaviour, it is possible to apply division and get the sequence of times $\{T_n\} = \{1, 2, \dots, n\}$. So when an evenly spaced time series is used, then it is written only as $\{X_n\}$ where $n \in \{1, 2, \dots, n\}$ and absolute observation times are unnecessary.

The evenly-spaced time series are often used in network traffic analysis, mainly for forecasting and anomaly detection [82,83]. Some previous works [59] use evenly spaced time series even for classification. However, network traffic naturally occurs with unevenly spaced timestamps (packet transmission time).

To create an evenly spaced time series from network traffic, we need to set the aggregation interval—the time window for a single data point in the series—that highly affects the analysis result due to packets occurring at the aggregation interval borders. Moreover, a majority of evenly-spaced time series from network traffic contains one or more intervals where no packet was transmitted—zero-value intervals. Badly selected aggregation intervals cause analysis failure: A large aggregation window causes a loss of information in behaviour patterns; a small aggregation window results in a large number of zero values, which highly affects the analysis results. Unfortunately, the ideal aggregation interval is different for each series and it can

[7] https://ieeexplore.ieee.org/
[8] https://dl.acm.org/
[9] https://www.scopus.com/
[10] https://scholar.google.com/

Table 1
Summary of datasets and related works for binary and multiclass classification. The Telemetry column represents the size of the flow extension (the classical flow has an extension size equal to zero). The Accuracy, F1-score, macro and weighted average F1-score are presented in percent [%].

Binary classification

Task	Dataset	Approach	Telemetry	Accuracy	F1-score
Botnet detection	CTU-13 [53]	Stergiopoulos et al. [54]	1,000	99.85	99.90
Brute-force detection	HTTPS Brute-force [55]	Luxemburk et al. [56]	180	99.93	96.26
Mining detection	CESNET-MINER22 [57]	Plný et al. [24]	680	93.72	90.59
DNS malware detection	CIC-Bell-DNS [15]	Kumaar et al. [58]	540	99.19	99.20
DoH detection	CIC-DoH-Brw [59]	Behnke et al. [60]	116	–	99.8
	DoH-Real-World [61]	Jeřábek et al. [13]	0	97.5	98.7
DoS attack detection	Bot-IoT [62]	Shafiq et al. [63]	176	99.99	99.99
IoT malware detection	IoT-23 [64]	Sahu et al. [65]	144	96	96
	Edge-IIoTset [66]	Khacha et al. [67]	472	99.99	99.99
	TON_IoT [68]	Dai et al. [69]	912	99.29	99.03
Intrusion detection	CIC-IDS-2017 [43]	Agrafiotis et al. [44]	3,136	98.5	95.4
	UNSW-NB15 [70]	Nawir et al. [71]	172	94.37	94.54
TOR detection	ISCX-Tor-2016 [72]	Dai et al. [69]	912	99.99	99.65
VPN detection	ISCX-VPN-2016 [73]	Aceto et al. [34]	1,296	93.75	91.95
	VNAT [74]	Jorgense et al. [74]	3,612	–	98.00

Multiclass classification

Task	Dataset	Approach	Telemetry	Accuracy	Macro avg. F1	Weighted avg. F1
Botnet classification	CTU-13 [53]	Marín et al. [75]	200	99.72	76.04	98.00
IoT malware classification	Edge-IIoTset [66]	Khacha et al. [67]	472	98.69	–	–
	TON_IoT [68]	Tareq et al. [76]	2,832	98.50	52.20	98.57
IDS classification	CIC-IDS-2017 [43]	Kunang et al. [77]	784	95.79	84.54	95.11
	UNSW-NB15 [70]	Madwanna et al. [18]	472	82.21	53.15	80.30
TOR classification	ISCX-Tor-2016 [72]	Dai et al. [69]	912	97.95	86.77	–
VPN classification	ISCX-VPN-2016 [73]	Dener et al. [78]	264	89.29	87.83	90.49
	VNAT [74]	Jorgense et al. [74]	3,612	96	–	–

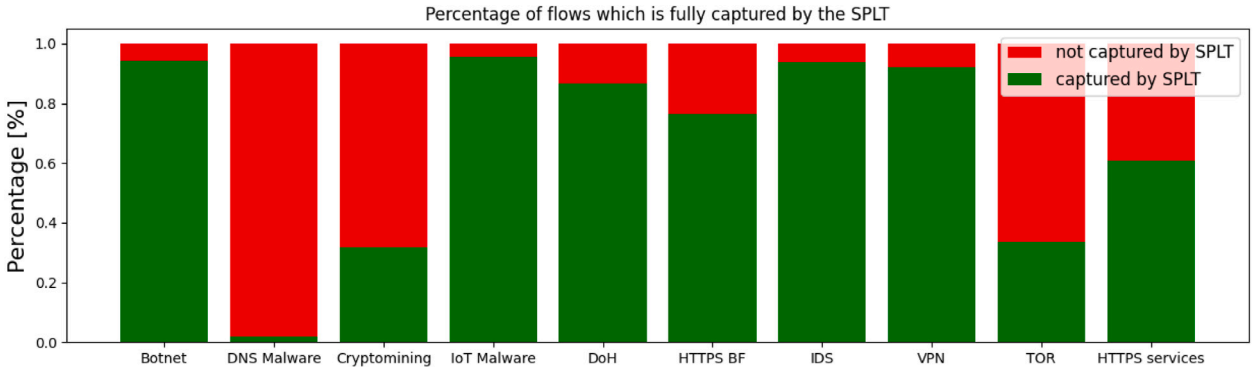


Fig. 1. The shares of network flows in selected tasks that can (cannot) be captured by SPLT of length 30.

also change in time—thus, the analysis failure with evenly spaced time series is inevitable [27].

However, the data can be used without aggregation. That means a time series of observations $\{X_n\} = \{x_1, x_2, \ldots, x_n\}$ taken at times $\{T_n\} = \{t_1, t_2, \ldots, t_n\}$ does not have constant $\delta_j = t_{j+1} - t_j, \forall j \in \{1, \ldots, n-1\}$. This type of time series is called unevenly (or unequally/ irregularly) spaced.

The unevenly-spaced time series from network traffic does not require any aggregation; moreover, they do not contain any trends and seasonality [27], which makes them more suitable for network traffic classification. Since there is already an advanced mathematical theory around unevenly-spaced time series (mainly in astronomy and medicine), we can use the mathematical methods and tools for their analysis to form a feature vector describing the properties of the communication and use it for network traffic classification [26]. Nevertheless, we also need to keep the computational and memory complexity in mind when designing the feature vector, to ensure the possibility of practical deployment.

4.2. Network traffic classification using time series analysis

The benefit of an unevenly spaced time series can be seen in Fig. 2, where each traffic type generates different patterns of packet sizes in time. These patterns can be captured by time series analysis [27]; thus, time series analysis features are an ideal candidate for a universal feature vector.

In our previously published work [26], we create a time series from packets transmitted within a single flow—the series of payload sizes in bytes with the corresponding transmission timestamp. We call them Single Flow Time Series (SFTS). The SFTS do not have evenly spaced timestamps between the data points; thus, they fall into the unevenly spaced time series category. Previously, we applied time series analysis on SFTS to generate a set of 69 features, which were then

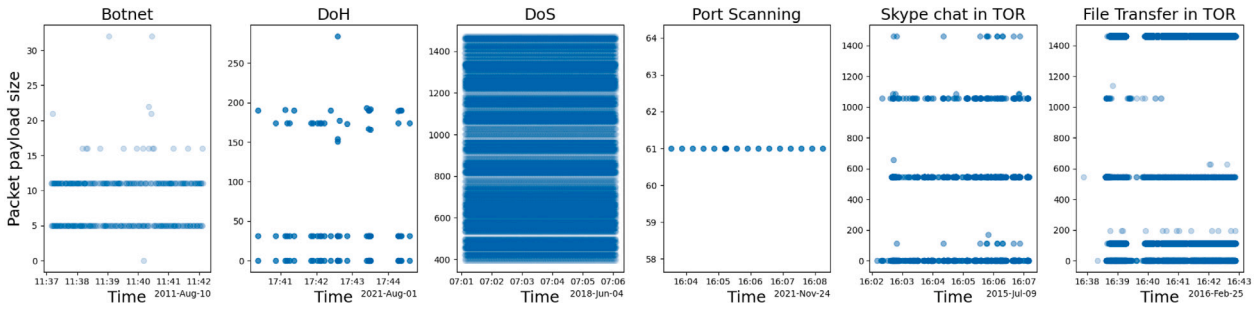


Fig. 2. Examples of packet time series of different traffic types.

tested for their discriminatory performance. Our evaluation involved 15 well-known publicly available datasets and showed that the proposed feature vectors achieve excellent classification performance in both binary and multiclass classification tasks.

Nevertheless, the previous work [26] did not consider deployment. Most of the features' computation complexity was not linear and required a whole time series saved in the memory. For example, the frequency-based features were derived from the Lomb–Scargle periodogram, which has computation complexity $O(n \log n)$. Additionally, some features require sorting the packet length sequence or computing complex statistical tests with quadratic complexity.

As a follow-up work to our previous paper, we further explored the possibility of optimization of time series analysis that would allow deployment in all network infrastructures, even the large and high-speed ones, by using online computation—a form of computation where continuous data are used to compute intermediate results without the need to save the whole data sequence. It usually requires only one or two values saved in memory. Nevertheless, not all statistical properties of time series can be computed exactly; in these cases, the use of approximation is necessary.

Therefore, similarly to previous work, we also create and analyse SFTS. However, in this paper, we propose features that can be extracted using online computation only—to save resources and minimize the performance impact. The exact features and their computation are described in the following section.

5. Features description

This section contains a detailed description of the novel feature set. It consists of three groups of complementary feature types to improve the robustness of the feature vector. The first group of features is based on classical bidirectional flow information—a number of transferred bytes, and packets. The second group contains statistical and time-based features calculated using the time-series analysis of the packet sequences. The bidirectional flow extended by the second group of features is called the NetTiSA flow. The third type of features can be computed from the previous groups (i.e., on the flow collector) and improve the classification performance without any impact on the telemetry bandwidth.

5.1. Flow features

In the flow features, we explicitly omit the use of IP addresses and transport ports since including them in the feature vectors is generally considered a bad practice, as described by Behnke et al. [60]. The flow features are:

Packets is the number of packets in the direction from the source to the destination IP address.

Packets in reverse direction is the number of packets in the direction from the destination to the source IP address.

Bytes is the payload size in bytes transferred in the direction from the source to the destination IP address.

Bytes in reverse direction is the payload size in bytes transferred in the direction from the destination to the source IP address.

5.2. Statistical and time-based features

These features are exported in the extended part of the flow, i.e., NetTiSA flow. All of them can be computed (exactly or approximately) by online computation, which is necessary for keeping memory requirements low. It contains the following features:

Mean represents the mean of the payload lengths of packets in the flow, computed with equation $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$.

Min is the minimal value of the payload lengths of all packets in the flow.

Max is the maximum value from payload lengths of all packets in the flow.

Standard deviation is a measure of the variation of payload lengths from the mean payload length. In general, it can be computed with equation $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \mu\right)^2}$. However, the mean value μ is not available for online computation. Therefore, we use an alternative equation $\sigma = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n} - \left(\frac{\sum_{i=1}^{n} x_i}{n}\right)^2}$.

Root mean square is the measure of the magnitude of payload lengths of packets computed as $rms = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$.

Average dispersion is the average absolute difference between each payload length of the packet and the mean value computed as $ad = \frac{1}{n}\sum_{i=1}^{n}|x_i - \mu|$. For the purpose of online computation, it must be computed with approximated mean $ad = \frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{\mu}_i|$ which is computed with equation $\hat{\mu}_i = \hat{\mu}_{i-1} + \frac{x_i - \hat{\mu}_{i-1}}{i}$, where $i := 1, 2, \ldots$ and $\hat{\mu}_0 := 0$. This equation allows computation in an online manner, without the need to store the whole payload sequence.

Kurtosis is the measure describing the extent to which the tails of a distribution differ from the tails of a normal distribution. It is computed by $Kurt = \frac{1}{n\sigma^4}\sum_{i=1}^{n}\left(x_i - \mu\right)^4$. For online computation, we will use the approximated mean as for the *Average dispersion*.

Mean of relative times is the mean of the relative times which is a sequence defined as $\{st\} = \{t_1 - t_1, t_2 - t_1, \ldots, t_n - t_1\}$.

Mean of time differences is the mean of the time differences of consecutive packets, which is a sequence defined as $\{dt\} = \{t_{i+1} - t_i \mid i \in \{1, 2, \ldots, n-1\}\}$.

Min of time differences is the minimal value of all the time differences, i.e., the shortest interval between packets.

Max of time differences is the maximum value of all the time differences, i.e., the longest interval between packets.

Time distribution describes the deviation of time differences between individual packets within the time series. The feature is computed by using the following equation:

$$tdist = \frac{\frac{1}{n-1}\sum_{i=1}^{n-1}\left|\mu_{\{dt\}} - dt_i\right|}{\frac{1}{2}\left(max\left(\{dt\}\right) - min\left(\{dt\}\right)\right)} \quad (1)$$

The maximum and minimum of the time differences are also calculated as a feature and the mean of the time differences is approximated the same way as for the mean calculated for the purpose of the *Average dispersion* and *Kurtosis* features. The lower the $tdist$, the better the time differences spread over time.

Switching ratio represents a value change ratio (switching) between payload lengths. The switching ratio is computed by $sr = \frac{s_n}{\frac{1}{2}(n-1)}$ where s_n is the number of switches, i.e., pairs of neighbouring packets with different lengths. Its online computation requires storing only two integer variables. One integer is for counting a number of different payload lengths of consecutive packets, and the second integer is for storing the last payload length value.

5.3. Features computed at the collector

The third set contains features 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 and contains the following features:

Max minus min is the difference between minimum and maximum payload lengths.

Percent deviation is the average dispersion normalized by the mean value, computed as $pd = \frac{ad}{\mu}$.

Variance is the spread measure of the data from its mean calculated as the square of the *Standard deviation*.

Burstiness is the degree of peakedness in the central part of the distribution computed as $B = \frac{\sigma-\mu}{\sigma+\mu}$.

Coefficient of variation compares the dispersion of a time series to its mean value. It is calculated as $cv = \frac{\sigma}{\mu}$.

Directions describe a percentage ratio of packet directions computed as $\frac{d_1}{d_1+d_0}$, where d_1 is a number of packets in a direction from source to destination IP address and d_0 the opposite direction. Both d_1 and d_0 are features of the classical bidirectional flow.

If they are all in the direction from the source to the destination, then the percentage is 100%; if they are all from the destination to the source, then the percentage is 0%.

Duration is the duration of the flow.

5.4. Feature extraction

We extracted flows with all proposed features from every PCAP from the datasets listed in Table 1. The flow extraction process was set with 300 s of active timeout and 65 s of inactive timeout.[11] Moreover, when a feature could not be computed (such as Standard deviation, or Switching ratio) due to a lack of data in short single-packet flows, we filled its value with "0". Finally, we split the datasets of flow data, extended by Enhanced NetTiSA features, into standard training, validation, and test parts.[12] Their detailed description is shown in Table 2. We made the created datasets publicly available using the Zenodo platform [28] to facilitate reproducibility.

6. Discriminative performance of features

In order to evaluate the performance of the proposed features for ML-based traffic classification, we trained a classifier for each of the datasets and tasks listed in Table 1 and evaluated the results. The performance of the classifiers then serves as a measure of the discriminative performance of the feature set.

Since the datasets have already been processed by NetTiSA feature extraction and split into train/validation/test parts (see Section 5.4), the classifier creation pipeline consists of only three steps: 1. Optimal ML algorithm selection, 2. Hyperparameter tuning, and finally 3. Model training and evaluation. The overall pipeline is shown in Fig. 3.

At first, we select the optimal ML algorithms. We tested 11 well-known ML algorithms, including kNN, SVM, Random Forest, XGBoost, and AdaBoost, from which the XGBoost algorithm outperforms all others on all classification tasks.

The XGBoost hyperparameters of the ML algorithms for each dataset and classification task were tuned using the validation dataset to avoid hyperparameter overfitting [84]. We use the *hyperopt library* [85] for tuning the following hyper-parameters: *n_estimators*, *max_depth*, *gamma*, *reg_alpha*, *reg_lambda*, *min_child_weight*, and *colsample_bytree*. The rest of the hyper-parameters were left at default values.

As the third step, we trained the best-performing XGBoost model with previously obtained hyperparameter values. The trained models were then evaluated on the test part of the dataset using standard classification metrics—*Accuracy* and *F1-score*, which are defined in the following formulas:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

$$Recall = \frac{TP}{TP + FN} \quad (4)$$

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (5)$$

Where TP is the number of *True positives*, TN is the number of *True negatives*, FP is the number of *False positives*, and FN is the number of *False negatives*. Furthermore, we use macro and weighted averages of these metrics for multiclass classification. The macro average is the arithmetic mean of scores computed for individual classes, while the weighted average also considers the support of each class. Nevertheless, these metrics often provide a simplified view of the performance of the classifiers, where some details (e.g., classification performance of a particular class) are lost. Therefore, we also provide confusion matrices in Appendix B.

[11] If 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.

[12] We used stratified sampling so that the original ratio of label classes remains in the split parts.

Table 2
Description of NetTiSA flow datasets created from original datasets.

Binary classification						
Task	**Original dataset**	**Labels [%]**		**Sizes of datasets**		
		False	**True**	**Train**	**Validation**	**Test**
Botnet detection	CTU-13 [53]	88.57	11.43	194,928	83,540	50,000
Mining detection	CESNET-MINER22 [57]	64.97	35.03	1,417,433	607,470	1,075,576
DNS Malware detection	CIC-Bell-DNS-2021 [15]	96.04	3.96	3,761	1,613	1,000
DoH detection	CIC-DoHBrw-2020 [59]	69.48	30.52	623,952	267,408	100,000
	DoH Real-world [61]	1.52	98.47	3,564,331	1,527,571	500,000
DoS detection	Bot-IoT [62]	97.40	2.59	2,106,874	902,946	1,000,000
Brute-force detection	HTTPS Brute-force [55]	95.71	4.29	646,366	277,014	100,000
IoT Malware detection	Edge-IIoTset [66]	92,95	7,05	827,557	354,668	250,000
	IoT-23 [64]	24.56	75.44	2,492,447	1,068,192	500,000
	TON_IoT [68]	0.31	99.69	2,077,190	890,224	500,000
Intrusion detection	CIC-IDS-2017 [43]	77.60	22.40	1,182,652	506,850	500,000
	UNSW-NB15 [70]	96.50	3.50	954,370	409,016	584,309
TOR detection	ISCX-Tor-2016 [72]	98.99	1.01	19,200	8,229	11,756
VPN detection	ISCX-VPN-2016 [73]	90.54	9.46	135,715	58,164	50,000
	VNAT [74]	96.61	3.39	23,380	10,020	10,000

Multiclass classification				
Task	**Original dataset**	**Sizes of datasets**		
		Train	**Validation**	**Test**
All binary classification[a]	[15,43,53,55,57,59,61, 62,64,66,68,70,72–74]	10,934,721	4,686,309	4,000,000
All multiclass classification[a]		7,127,894	3,054,812	9,999,803
Botnet classification	CTU-13 [53]	106,212	45,519	25,000
Intrusion classification	CIC-IDS-2017 [43]	1,392,651	596,851	200,000
	UNSW-NB15 [70]	954,370	409,016	584,309
IoT Malware classification	Edge-IIoTset [66]	827,557	354,668	250,000
	TON_IoT [68]	2,252,190	965,224	250,000
TOR classification	ISCX-Tor-2016 [72]	70,557	30,239	20,000
VPN classification	ISCX-VPN-2016 [73]	12,655	5,423	5,000
	VNAT [74]	23,380	10,020	10,000

[a] Task was created by merging all relevant datasets (see Section 6.1.3).

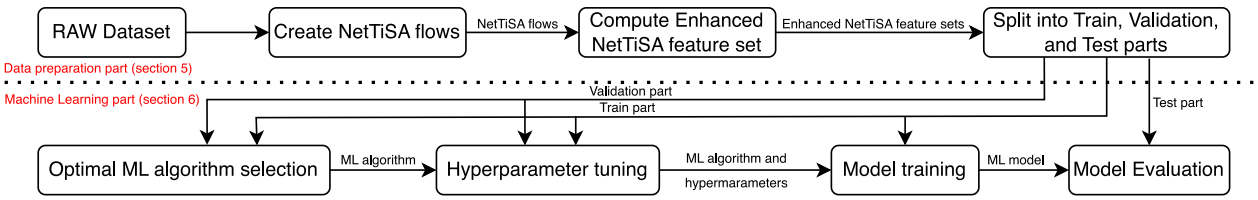


Fig. 3. Experimental process for design and evaluation of ML algorithms trained using NetTiSA.

The test part was not used during any stage of the classifier design, ensuring the fairness of model evaluation on data that was not seen before. The source codes of our whole classification pipeline, including the pre-processing and the final settings of the hyperparameters for each classification problem, are publicly available in our repository.[13]

6.1. Classification results

In this section, we present classification performance on 25 network classification tasks using 15 publicly available and well-known network traffic datasets (listed in Table 2). The tasks can be separated into 15 binary classification tasks, 8 multiclass classification tasks, and two special multiclass classification tasks. The performance of the classifier trained on the novel Enhanced NetTiSA features is always compared with the best-performing classifier from the related work using the previously defined metrics. For each metric, a higher value means better performance of the classifier; both accuracy and F1 score metrics are considered in the evaluation separately. Moreover, we also compare the performance with our previously published results (TSA of SFTS) [26], to show the comparison between two types of universal time-series-based features.

6.1.1. Binary classification

The results of binary classification are graphically presented in Fig. 4. We also provide detailed results in Appendix, in Tables A.8 and B.10.

The classification based on the Enhanced NetTiSA feature set achieved mostly similar or better results than the best-performing classifier from related works.[14] On six binary classification tasks, the Enhanced NetTiSA features performed significantly better—it achieved more than

[13] https://github.com/koumajos/Classification_by_NetTiSA_flow

[14] The best-performing related work for each dataset is listed in Table 1.

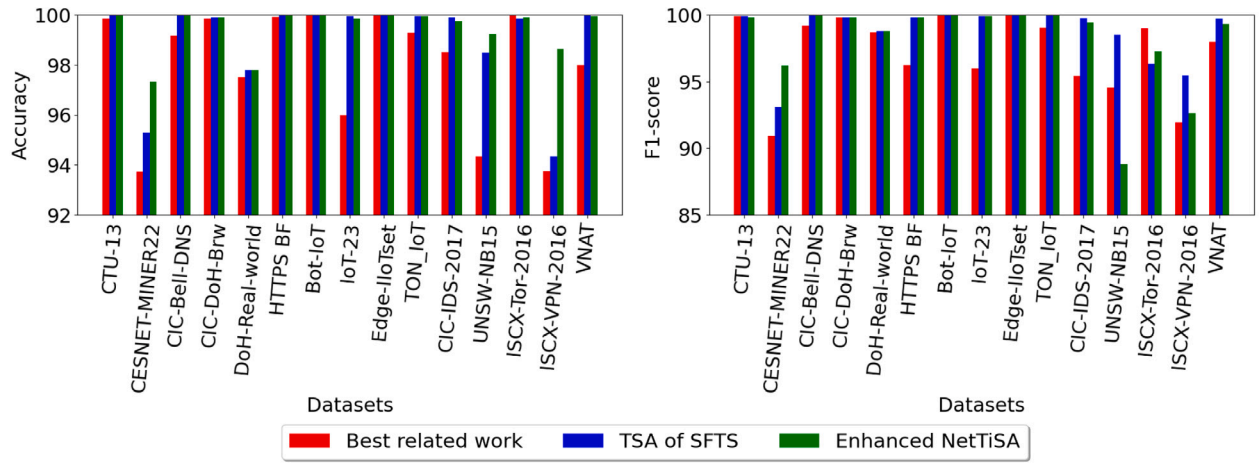


Fig. 4. Comparison of binary classification by best-related works, complete feature set based on TSA of SFTS and Enhanced NetTiSA feature set. We compare approaches by Accuracy and F1-score (in %).

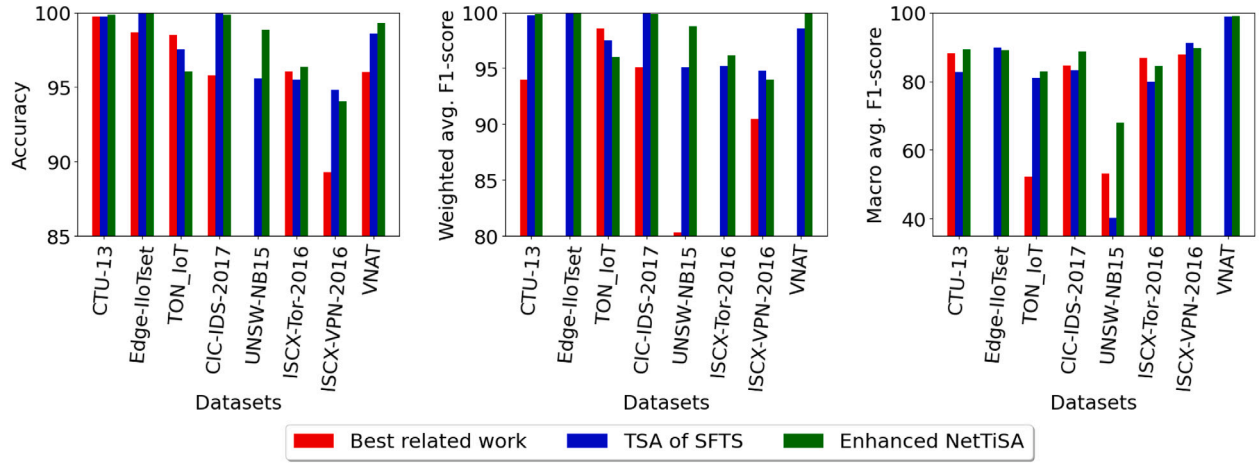


Fig. 5. Comparison of multiclass classification by best-related works, complete feature set based on TSA of SFTS and Enhanced NetTiSA feature set. We compare approaches by Accuracy, and macro and weighted average F1-score (in %).

1% increase in accuracy measure or F1 score compared to related works. Our approach achieves more than 1% increase in problems of detecting cryptomining, IoT malware, intrusion, and VPN. Especially, the detection of crypto mining using the CESNET-MINER22 dataset and VPN using the ISCX-VPN-2016 dataset achieves a better F1 score by more than 5%. Nevertheless, with the TOR detection problem and Intrusion detection on the UNSW-NB-15 dataset, we observed a slightly worse F1 score than the best-performing classifier from the related works.

We investigated the related work of Intrusion detection on the UNSW-NB-15 dataset. The classifier published by Nawir et al. [71] indeed performed significantly better by the means of F1-score (by 5.75%), but their accuracy is significantly worse than that of our approach (by 4.85%). The increased F1 score can be attributed to the application-layer-based features (such as from HTTP headers) that would not be applicable in encrypted traffic analysis.

We also investigated the difference between the TOR classifier published by Dai et al. [69], which extends flow data for the first 32 packets and also 600 bytes of payload. The payload information gives their classifier an advantage since the classifier can fit the unencrypted TLS handshake data. Since TOR usually does not use some common TLS extensions, e.g., Server Name Indication, the TOR TLS handshakes in the dataset are specific and easily recognizable by payload-based analysis. The raw payload data gives Dai et al. an advantage, but also result in much larger flow records (as shown in the next section).

When we compare the Enhanced NetTiSA discriminative performance with universal Time-Series Analysis features (TSA of SFTS) from our previous work [26], we can see (in Fig. 4) that the results are mostly similar. The TSA of SFTS features show significantly better discriminative performance (when concerning the F1-score) in the ISCX-VPN-2016 and UNSW-NB-15 datasets. On the other hand, the NetTiSA is significantly better on the CESNET-MINER22 dataset. The overall comparison of the binary classification is summarized in Table 3.

6.1.2. *Multiclass classification*

The results of multiclass classification are presented in Fig. 5. Furthermore, the detailed results are also presented in the Appendix, Tables A.9, B.11, B.12, B.13, B.14, B.15, B.16, B.17, and B.18. 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 (listed in Table 1). Specifically, in five out of eight cases (botnet families, IoT malware families, intrusion type, and VPN traffic type), we achieved more than a 1% classification performance increase compared to best-performing related works. Especially, the classification of intrusion type with NetTiSA features extracted from the UNSW-NB15 dataset achieves a better F1 score by more than 15%. However, we observed a slight decrease in two cases—TON_IoT and TOR classification.

Table 3
Comparison of average results of binary and multiclass classification based on NetTiSA flows with classification based on all features generated using TSA of SFTS. Comparison is made by the average and standard deviation of each classification metric across all datasets (in %).

	Binary classification		Multiclass classification		
	Accuracy	F1-score	**Accuracy**	**Weighted average F1-score**	**Macro average F1-score**
NetTiSA	99.48 (±0.840)	98.13 (±3.167)	98.03 (±2.098)	86.34 (±8.267)	98.06 (±2.186)
TSA of SFTS	99.02 (±1.772)	98.74 (±2.026)	97.70 (±2.025)	80.87 (±16.480)	97.59 (±2.140)
Difference	0.45 (±1.154)	−0.61 (±2.681)	0.32 (±1.308)	5.47 (±8.845)	0.46 (±1.477)

Table 4
Final results (in %) from the testing phase of multiclass classification based on NetTiSA flow of all binary problems together.

Class	**Precision**	**Recall**	**F1-score**	**Support**
Botnet	99.36	96.66	97.99	12,751
Clear	97.55	96.82	97.18	5,283,309
Cryptomining	99.73	99.52	99.62	368,528
DNS malware	90.79	76.67	83.13	90
DoH	92.39	94.98	93.67	1,844,524
DoS	98.70	91.33	94.87	20,711
IoT malware	99.80	99.23	99.51	13,287
HTTPS Brute Force	93.60	89.89	91.70	200,484
Intrusion	98.03	98.06	98.05	2 247,509
TOR	85.80	65.07	74.01	418
VPN	93.71	71.25	80.95	8,389
Accuracy	–	–	96.69	10,000,000
Macro average	95.41	89.04	91.88	10,000,000
Weighted average	96.71	96.69	96.69	10,000,000

Table 5
Final results (in %) from the testing phase of multiclass classification based on NetTiSA flow of all multiclass problems together. The crated dataset of all multiclass problems contains 44 classes.

	Accuracy	**Precision**	**Recall**	**F1-score**	**Support**
Macro average	95.08	83.49	66.51	71.81	9,999,803
Weighted average	95.08	95.03	95.08	94.99	9,999,803

The best-performing classifier of TON_IoT published by Tareq et al. [76] is based on a 2D convolutional network (CNN) with very long packet-length data (SPLT with all packets from connection) organized in the image. The SPLT data give the classifier an advantage in the opportunity of high-quality feature extraction that allows accurate classification. Nevertheless, the long packet sequences (SPLT) cannot be exported in real-world deployment scenarios due to the technical limitations of the flow exporters (see Section 2). Furthermore, although the Tareq et al. [76] 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 Enhanced NetTiSA feature set. Thus, their classifier is unable to classify some less prevalent classes reliably.

Similarly, as in the binary classification of TOR, Dai et al. [69] achieved a better Macro average F1-score of 86.77%, while the classifier trained with NetTiSA achieved only 79.87%. The reason is the same—the raw payload data included in the flow records.

When we compare the Enhanced NetTiSA with universal Time-Series Analysis features (TSA of SFTS) from our previous work [26], we can see 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 and Weighted average F1-score) than our previous work. The overall comparison with our previous work is shown in Table 3. It can be seen that on multiclass classification, on average, the NetTiSA performs better in all performance metrics.

6.1.3. Multiclass classification using merged datasets

Since the Enhanced NetTiSA feature vector performed well in all evaluated cases—either binary or multiclass, we wanted to challenge the discriminative performance even more. Therefore, we merged the datasets to create a more complex problem with more classes. At first, we experiment with merging all datasets used for the binary classification problems. The resulting dataset contains ten classes.[15] The results of classification are presented in Table 4. We achieve 97.14% of the weighted average F1-score and 91.88% of the macro average F1-score.

Moreover, we merged all multiclass problems with several binary problems (detection of cryptomining, DNS malware, DoH, HTTPS Brute Force) into a single dataset. The created dataset contains 44 classification classes (see Appendix C). The achieved results are shown in Table 5. The classifier achieved 94.99% of the weighted average F1-score and 71.81% of the macro average F1-score.

In both cases, the classifier performed well despite the large imbalance of the classes in the dataset—which explains the lower performance in the macro average F1-score. Nevertheless, the results show that the NetTiSA features achieve high accuracy even on fine-grained classification tasks with many classes.

6.2. Enhanced NetTiSA feature importances

We extracted the feature importances of individual Enhanced NetTiSA features measured by the gain metric[16] from each trained model. Nevertheless, the importance differs widely on each evaluated classification task; therefore we average the values across all cases. The average feature importances are presented in Fig. 6. In the top ten most important features, there are mainly time-based features.

Some classification tasks strongly influenced the feature importance of a single particular feature. For example, the high importance of the feature *Directions* is caused by the Botnet, (D)DoS, IDS using the CIC dataset, and VPN (VNAT dataset) binary classification problems. Since in these problems the asymmetry of the traffic is extremely important

[15] Botnet, Clear, Cryptomining, DNS Malware, DoH, DoS, IoT Malware, HTTPS Brute Force, Tor, and VPN.

[16] The average gain across all splits the feature is used in.

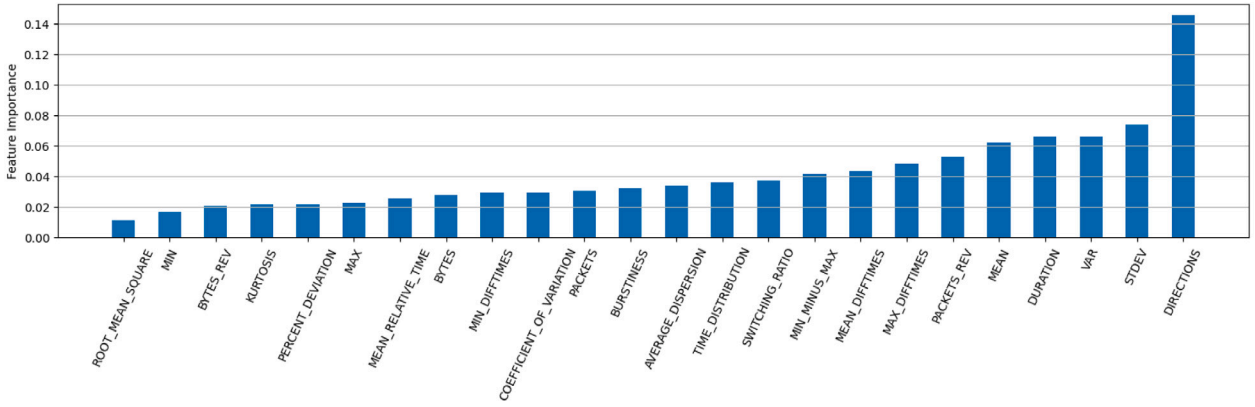


Fig. 6. Average feature importance across all classification problems measured by gain importance metric.

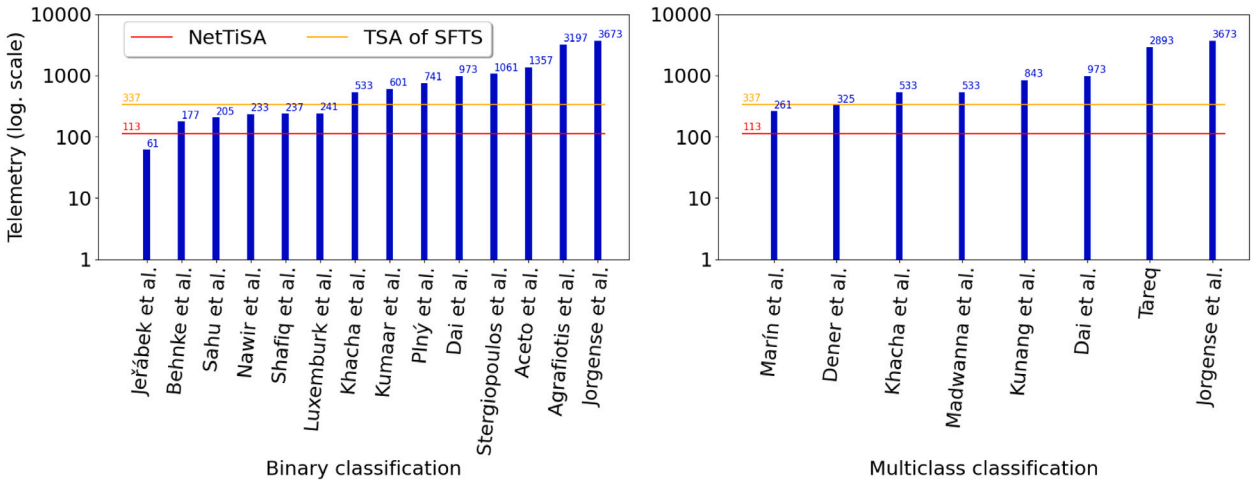


Fig. 7. Comparison of telemetry size of best-related works, complete feature set based on Time Series Analysis of Single Flow Time Series (TSA of SFTS) from our previous work [26], and NetTiSA flow.

for classification, the *Directions* feature separates the majority of the dataset samples.

Other binary classification problems use primarily one or two features. Nevertheless, all multiclass problems use almost all features with small differences in their importance. The detailed view on feature importances for each classification task is shown in Appendix D.

6.3. Discussion about feature discriminative performance

Despite the expected performance drops, due to the universality of the feature set, we consider the Discriminative Performance of Enhanced NetTiSA features as very good and comparable with the best-performing proposals from the state-of-the-art. In only four cases out of 23 the classifier performed more than 1% worse than the related work, while 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.

7. Network telemetry size comparison

The size of the network telemetry matters, especially in commercial deployment. A larger flow extension puts additional stress on the monitoring infrastructure—the collectors and detectors must process more data. Moreover, in the case of flow monitoring infrastructure [11], large telemetry also utilizes bandwidth that could be otherwise used by customers. Therefore, we compared the sizes of an extended part of flow records used as the input for the classification across all the concerned tasks.

To form a reference point, we need to establish the size of a classical IP flow record. We define such flow as a combination of flow key (Source IP, Destination IP, Source port, Destination port, Protocol) and a tuple of 6 values (number of packets and bytes in the flow in both directions, first and last packet timestamp). Thus, the transfer of this classical IP flow requires 21 bytes for the flow key and 40 bytes for the 6-value tuple—resulting in 61 bytes.

Most of the approaches extend the flow for additional data, which naturally results in the size increase. We analysed the flow records used by the best-performing classifiers on each evaluated classification task and established their input flow record size. When the datatypes were not specified, we assumed four-byte integers and four-byte floats for the record size calculation.

The results of our analysis are presented in Fig. 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 two times larger flow extension. The only approach that uses smaller flow records is the Jeřábek et al. [13], who compensate for the small amount of information with active probing.

7.1. Discussion about the telemetry size

The Fig. 7 also shows that most approaches do not consider a telemetry size as an important factor. For example, the Jorgense

et al. [74] on the VNAT dataset requires 3673 bytes on a single flow record—their records would consume around 29 Gb of bandwidth per million of flows. Such bandwidth requirement makes the approach hardly deployable since the large ISP network can generate more than a million of 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, we consider its telemetry size acceptable even in large infrastructures.

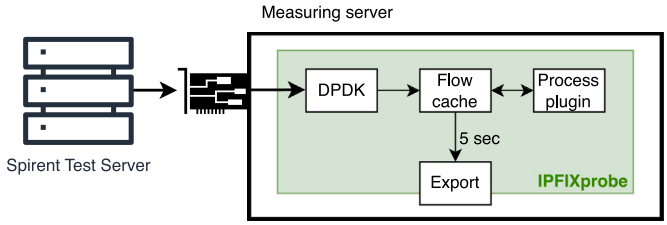


Fig. 8. Measuring setup.

8. The impact of NetTiSA features calculation

To prove the feasibility of NetTiSA features in high-speed infrastructures, we also discuss the computational efficiency and measure the performance impact of NetTiSA flow features calculation. We implemented the feature calculation into flow exporter ipfixprobe,[17] which serves as a reference implementation of the novel feature set. The ipfixprobe is a high-performance flow exporter written in C++, that is capable of monitoring 100 Gbps backbone lines, for example, in the CESNET2 network. It supports multiple input types, from traditional slow interfaces such as libpcap to high-speed DPDK (Data Plane Development Kit) [86] inputs for ISP-based monitoring.

8.1. Computational efficiency of NetTiSA

Regarding computational efficiency, the basic bidirectional flow measurement process is linear with the number of packets in the flow n, denoted as $\mathcal{O}(n)$ [11]. The NetTiSA feature vector executes a constant number of operations. Analysing these operations, NetTiSA

- does not include cycles, uses only if-else branching,
- performs arithmetic operations in memory,
- uses mathematical functions `abs`, `pow`.

The only discussable problem is using function `pow`. However, in the NetTiSA definition, the exponentiation is performed with exponents 2 and 4 independently on the number of packets; thus, even for `pow`, the complexity is constant, $\mathcal{O}(1)$. This implies that the overall computational complexity will be in the same class as the computation of basic bidirectional flows, i.e. linear with the number of packets, $\mathcal{O}(n)$.

8.2. Experiments

We leveraged the plugin-based architecture of the ipfixprobe and implemented NetTiSA flow as the process plugin with the same name. The plugin performs the online computation of each feature described in Section 5 for each flow.

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

[17] https://github.com/CESNET/ipfixprobe

For the comparison, we measured the ipfixprobe exporter 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. full set of Time Series Analysis features (TSA of SFTS) [26], which we also implemented as an additional ipfixprobe process plugin. The plugins have been tested in two environments. In the lab-traffic environment, we performed a stress test and tested the limits of the exporters using artificially created data. In the real-traffic environment, we utilized anonymized pcaps captured at the backbone ISP network and tested the exporters using real network data.

8.3. Performance with lab-traffic environment

First, we measure the performance with the lab-traffic environment. For this purpose, we utilized the Spirent test server. The Spirent test server generates the network traffic according to the specified parameters and thus performs stress tests of network devices. The packets were transmitted to a server with ipfixprobe exporter, instantiated with the DPDK input plugin to allow high-speed packet processing. The measuring setup is shown in Fig. 8.

As mentioned above, we are only interested in the overall performance depending on flow length. Therefore, we defined three constraints to remove the influence of the input part of the exporter and flow cache as much as possible:

1. **Used uniformly randomized packet sizes** from 80 to 1400 bytes to maintain variable data for the processing.
2. **Limited the traffic to only a single network flow.** The single flow stress test removes the influence of other parts in the exporter (such as cache misses, and collisions) and allows us to focus on the processing plugin performance and memory requirements. We force the flow cache to flush every five seconds to ensure that more than one flow is exported during the measurement. The measurement time is always 60 s; thus, the count of the exported flows should be exactly 12 ($60/5 = 12$). The number of packets/bytes in flow is then regulated by setting frames generated per second.
3. **Set only one input DPDK thread** since we measured in the environment where a specific single flow is always mapped to the same DPDK queue. Thus, using multiple threats would be redundant.

8.3.1. Computational performance

The computational performance of the evaluated plugins is shown in Fig. 9. In the top left chart, we show the maximal throughput of the exporter based on the used plugin. The exporter is saturated relatively early in terms of frames per second—the throughput of the exporter is approximately 500,000 frames per second (due to applied limitations described in Section 8.3). After that, we do not observe any throughput gain. However, we can see that the TSA of SFTS causes the saturation much earlier, and we cannot get a better throughput than around 200 Mbps.

In the top right chart, we measured the delay between the timeout of the flow and the actual export of the flow record. As we can see, in almost all cases, this delay is about 0.7 s, which is probably caused by buffering in ipfixprobe. Only for TSA of SFTS the delay is significantly longer for higher frame rates, which is caused by complex computations performed when a flow is to be exported.

The bottom left chart is a complement to the second chart. It shows how many flows the plugin can export—the maximum is 12, which means only the TSA of SFTS cannot export all available flows when packet per second are equal to or larger than 100,000.

Overall, Fig. 9 shows that the computation speed of NetTiSA and SPLT features are comparable to classic IP Flow, and we do not observe any performance drop. On the other hand, the plugin TSA of SFTS lags behind (performance-wise) in every perspective.

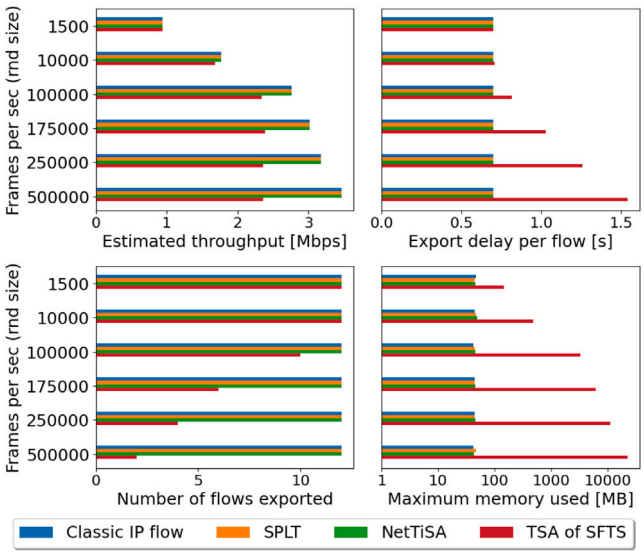


Fig. 9. Results of performance and memory requirements comparison using Spirent test server.

8.3.2. Memory requirements

In the bottom right chart of Fig. 9 we see the memory footprint of 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 is included. Since the processed data was identical, this overhead can be assumed to be comparable and thus detectable in the whole measurement. We can observe that computation of SPLT and NetTiSA flows requires small amounts of additional memory, while TSA of SFTS needs extreme amounts, which increase with the number of packets in the flow.

8.4. Performance on captured real traffic

For performance measurement on captured real traffic, we used three types of PCAP files 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 real office environment of a subnetwork inside ISP CESNET2 network.

8.4.1. Computational performance

The performance comparison by the means of processing time needed for each feature type is shown in Table 6. It must be noted that the processing time is influenced by loading packets from the PCAP file (the processing time of the same number of packets received directly from the network would be much faster). We can observe expected behaviour in performance on the real traffic captures. The Classical IP flow has the best performance because it performs the least operations, 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 SFTS needs a lot of computation, resulting in a significant decrease in performance compared to other types of flows.

8.4.2. Memory requirements

The results of the memory requirement of each set of feature computations are shown in Table 7. The results are provided as the mean and standard deviation of memory usage measured every second. We can see several points from the results: 1. The number of packets in flow does not influence the memory requirements of the feature computation of Classical IP, SPLT, and NetTiSA flows. 2. The number of flows does not highly influence the memory requirements of the feature computation of Classical IP and NetTiSA flows. However, the SPLT flow is influenced by it. 3. The TSA of SFTS 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 SFTS is highly unstable because of freeing large vectors of payload lengths and times after export of the flow. 5. The difference between the memory requirements of the TSA of SFTS on Short and Long flows is huge (from 26.5 MiB on 1k of flows to 5.24 GiB on 1k of flows).

Overall, the computation of NetTiSA flows requires only a small amount of memory, which is stable across different measured cases.

8.5. Deployment of the NetTiSA flow

Since the NetTiSA feature computation performance was almost identical to the traditional flow computation, we consider its performance excellent. Moreover, the NetTiSA plugin has already been deployed in the production ISP network CESNET2 with eight monitoring probes, with approximately 200 thousand flows each second. So far, the NetTiSA flows are used to create novel datasets from the CESNET2 network with encrypted traffic. Creating datasets with NetTiSA features represents the first step in our planned deployment of network threat detection and encrypted traffic annotation using machine learning.

The deployment of the NetTiSA feature export into other flow exporters than ipfixprobe requires minor changes in the flow property computation. The developers can take advantage of the reference solution. Moreover, we plan to create packet processing plugins to popular flow monitoring software such as Zeek,[18] or Argus[19] to bring the NetTiSA to the community. As shown in this section, including the novel features does not significantly affect the existing flow exporter performance.

Based on the performance results provided in this section, we consider the NetTiSA as deployment-ready to a wide range of monitoring infrastructures—from small home or office networks to ISP-based backbone lines, where millions of users can benefit from its high discriminative performance in various network security detection tasks.

9. Conclusion

In this paper, we proposed a novel extended IP flow called NetTiSA flow built on Time Series Analysis of Single Flow Time Series. The NetTiSA flow contains 13 features based on statistics of payload lengths, statistics of transmission times. Moreover, additional seven features can be computed from NetTiSA and traditional flow features, resulting in 20 features called Enhanced NetTiSA. The purpose of additional features is to improve the classification performance. The usability of Enhanced NetTiSA feature set was thoroughly evaluated from three main perspectives: 1. Discriminative performance, 2. Flow telemetry size and bandwidth requirements, and 3. Feature computational overhead.

The discriminative performance of the Enhanced NetTiSA feature set was evaluated on 25 network classification tasks using 15 publicly available and well-known network traffic datasets. These datasets were used to train and evaluate ML models and compare their performance to the best-performing classifiers from related works. In almost all cases, a model using the Enhanced NetTiSA feature set gives similar or better results than the best performing previous proposal for the corresponding task. This confirms an excellent discriminative performance and universality of the proposed features.

[18] https://zeek.org
[19] https://openargus.org

Table 6
Performance comparison on real traffic by using captured PCAPs.

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

Table 7
The comparison of memory usage per second of input PCAPs processing. The comparison is done by mean and standard deviation.

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

At the same time, the size of the NetTiSA flow extension remains small. Even on a network with 1 million flows per second, flows can be exported via a single 1 Gbps line; as our experiments showed, computation of NetTiSA features has only negligible impact on the performance of the flow monitoring probe. However, it can still process 100 Gbps of network traffic. Together, these properties make NetTiSA easy to deploy even in large ISP networks. Indeed, we already use the NetTiSA flow extension within the production monitoring infrastructure of the CESNET2 network.

We made the implementation of NetTiSA computation open-source as part of the *ipfixprobe* flow exporter. Moreover, the implementation of the feature extraction is developed in the form of a library, which allows fast integration into other flow-monitoring software.

Overall, the novel Enhanced NetTiSA feature set has the potential to replace the current de facto standard SPLT and other flow extension approaches due to its lightweight extraction, small flow size, and excellent discrimination capability across multiple network traffic classification tasks. We consider the NetTiSA as deployment-ready to a wide range of monitoring infrastructures—from small home or office networks to ISP-based backbone lines, where millions of users can benefit from its high discriminative performance in various network security detection tasks.

9.1. Future work

As a future work, we plan to integrate NetTiSA into other network monitoring software to bring the features closer to the community. Nevertheless, we also identified multiple research tasks that should be addressed in the future.

Fine-grained classification The NetTiSA feature set achieved excellent discriminative performance on multiclass classification. However, the used standard networking datasets contain only a handful of classification classes. Recently, novel datasets, such as CESNET-TLS22 [16] or CESNET-QUIC22 [80] emerged, offering hundreds of classes, making the classification task more challenging but also closer to the real-world needs. Nevertheless, these datasets cannot be directly used with NetTiSA, since they contain only short SPLT sequences. Thus, the first step involves creating novel fine-grained datasets with raw packet data, allowing NetTiSA feature calculation.

Combination with SPLT The SPLT features are de-facto standard for encrypted traffic classification. They are designed to work mainly for short flows since, on longer flows, the SPLT of limited length loses a large amount of information. The combined usage of SPLT and NetTiSA, which works better on longer flows, can bring an additional benefit of increased classification accuracy.

Low-rate attack The evaluation of the NetTiSA features has been done on the standard networking datasets. Nevertheless, they do not contain all types of traffic attacks, especially the low-rate ones. The evaluation of discriminative performance on attacks using low and slow strategies should be done in the future since these types of attacks are more challenging to detect.

CRediT authorship contribution statement

Josef Koumar: Conceptualization, Data curation, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **Karel Hynek:** Writing – original draft, Conceptualization, Methodology, Validation. **Jaroslav Pešek:** Data curation, Investigation, Software, Writing – original draft. **Tomáš Čejka:** Funding acquisition, Project administration, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The used dataset is available at [28].

Acknowledgements

This research was funded by the Ministry of Interior of the Czech Republic, grant No. VJ02010024: Flow-Based Encrypted Traffic Analysis and also by the Grant Agency of the CTU in Prague, grant No. SGS23/207/OHK3/3T/18 funded by the MEYS of the Czech Republic.

Appendix A. Complete results of classification based on Enhanced NetTiSA feature set

The detailed performance results of classification for all evaluated tasks can be seen in Tables A.8 and A.9.

Table A.8
Comparison of results of binary classification based on Enhanced NetTiSA feature set with best-related work on the same dataset. The Telemetry column represents the size of the flow extension (the classical flow has an extension size equal to zero). The green background colour marks fields where classification based on the NetTiSA flow is at least 1% better than best-related work. The red background colour marks fields where classification based on the NetTiSA flow is at least 1% worse than best-related work. The grey background colour marks the rest fields.

Task	Dataset	Approach	Telelemetry	Accuracy	F1-score
Botnet detection	CTU-13 [53]	Stergiopoulos et al. [54]	1,000	99.85	99.90
		TSA of SFTS [26]	276	99.98	99.93
		NetTiSA	**52**	99.95	99.79
Mining detection	CESNET-MINER22 [57]	Plný et al. [24]	680	93.72	90.59
		TSA of SFTS [26]	276	95.29	93.11
		NetTiSA	**52**	**97.32**	**96.19**
DNS Malware detection	CIC-Bell-DNS [15]	Kumaar et al. [58]	540	99.19	99.20
		TSA of SFTS [26]	276	100.0	100.0
		NetTiSA	**52**	100.0	100.0
DoH detection	CIC-DoH-Brw [59]	Behnke et al. [60]	116	–	99.8
		TSA of SFTS [26]	276	99.90	99.84
		NetTiSA	**52**	99.89	99.83
	DoH-Real-world [61]	Jeřábek et al. [13]	**0**	97.5	98.7
		TSA of SFTS [26]	276	97.79	98.80
		NetTiSA	52	97.79	98.80
Brute force detection	HTTPS Brute-force [55]	Luxemburk et al. [56]	180	99.93	96.26
		TSA of SFTS [26]	276	99.99	**99.83**
		NetTiSA	**52**	99.98	**99.80**
DoS detection	Bot-IoT [62]	Shafiq et al. [63]	176	99.99	99.99
		TSA of SFTS [26]	276	100.0	100.0
		NetTiSA	**52**	100.0	99.98
IoT Malware detection	IoT-23 [64]	Sahu et al. [65]	144	96	96
		TSA of SFTS [26]	276	**99.86**	**99.91**
		NetTiSA	**52**	**99.85**	**99.90**
	Edge-IIoTset [66]	Khacha et al. [67]	472	99.99	99.99
		TSA of SFTS [26]	276	99.99	99.97
		NetTiSA	**52**	100.0	99.98
	TON_IoT [68]	Dai et al. [69]	912	99.29	99.03
		TSA of SFTS [26]	276	99.96	99.98
		NetTiSA	**52**	99.95	99.97
Intrusion detection	CIC-IDS-2017 [43]	Agrafiotis et al. [44]	3,136	98.5	95.4
		TSA of SFTS [26]	276	**99.89**	**99.75**
		NetTiSA	**52**	**99.75**	**99.43**
	UNSW-NB15 [70]	Nawir et al. [71]	172	94.37	94.54
		TSA of SFTS [26]	276	**98.48**	**98.50**
		NetTiSA	**52**	**99.22**	88.79
TOR detection	ISCX-Tor-2016 [72]	Dai et al. [69]	912	99.99	**99.65**
		TSA of SFTS [26]	276	99.84	96.33
		NetTiSA	**52**	99.91	97.24
VPN detection	ISCX-VPN-2016 [73]	Aceto et al. [34]	1,296	93.75	91.95
		TSA of SFTS [26]	276	94.35	**95.48**
		NetTiSA	**52**	**98.64**	92.64
	VNAT [74]	Jorgense et al. [74]	3,612	–	98.00
		TSA of SFTS [26]	276	**99.98**	**99.73**
		NetTiSA	**52**	**99.96**	**99.35**

Table A.9
Comparison of results of multiclass classification based on Enhanced NetTiSA feature set with best-related work on the same dataset. We compare results by Accuracy in %, Macro average F1-score in %, and Weighted average F1-score in %. The Telemetry column represents the size of the flow extension (the classical flow has an extension size equal to zero). The green background colour marks fields where classification based on the NetTiSA flow is at least 1% better than the best-related work. The red background colour marks fields where classification based on the NetTiSA flow is at least 1% worse than the best-related work. The grey background colour marks the rest fields.

Task	Dataset	Approach	Telelemetry	Accuracy	Macro average F1	Weighted average F1
Botnet classification	CTU-13 [53]	Marín et al. [75]	200	99.72	76.04	98.00
		TSA of SFTS [26]	276	99.73	82.79	99.73
		NetTiSA	**52**	99.85	**89.31**	**99.84**
IoT Malware classification	Edge-IIoTset [66]	Khacha et al. [67]	472	98.69	–	–
		TSA of SFTS [26]	276	**99.97**	**89.75**	**99.97**
		NetTiSA	**52**	**99.96**	**89.01**	**99.96**
	TON_ IoT [68]	Tareq et al. [76]	2,832	**98.50**	52.20	**98.57**
		TSA of SFTS [26]	276	97.53	81.02	97.51
		NetTiSA	**52**	96.06	**82.82**	96.01
Intrusion classification	CIC-IDS-2017 [43]	Kunang et al. [77]	784	95.79	84.54	95.11
		TSA of SFTS [26]	276	**99.93**	83.23	**99.92**
		NetTiSA	**52**	**99.86**	**88.70**	**99.85**
	UNSW-NB15 [70]	Madwanna et al. [18]	472	82.21	53.15	80.30
		TSA of SFTS [26]	276	95.60	40.22	95.08
		NetTiSA	**52**	**98.85**	**67.90**	**98.78**
TOR classification	ISCX-Tor-2016 [72]	Dai et al. [69]	912	**97.95**	**86.77**	–
		TSA of SFTS [26]	276	95.48	79.87	95.20
		NetTiSA	**52**	96.36	84.46	96.16
VPN classification	ISCX-VPN-2016 [73]	Dener et al. [78]	264	89.29	87.83	90.49
		TSA of SFTS [26]	276	**94.80**	**91.21**	**94.77**
		NetTiSA	**52**	**94.04**	89.66	**94.00**
	VNAT [74]	Jorgense et al. [74]	3,612	96	–	–
		TSA of SFTS [26]	276	**98.60**	**98.88**	**98.60**
		NetTiSA	**52**	**99.29**	**98.92**	**99.29**

Appendix B. Confusion matrices

The confusion matrices for all evaluated classificaiton tasks are provided in Tables B.10–B.18.

Table B.10
Confusion matrix for each binary classification problem. The first row represents the "False" labels, and the second row represents the "True" labels in the dataset. The first column represents predicted "False", and the second column is predicted "True" values by the classifier.

Botnet		Cryptomining		DoH – RealWorld		DoH – CIC		DNS Malware	
23,398	4	682,969	30	26,815	14,006	69,516	28	959	1
2	1,596	28,812	363,765	2,103	457,076	78	30,378	5	35
DoS		**IoT – Edge-IIoT**		**HTTPS Brute-force**		**IDS – CIC**		**IDS – UNSW**	
984,942	0	232,541	4	95,663	3	287,752	546	561,766	2,099
6	15,052	4	17,451	14	4,320	721	110,981	2,457	17,987
TOR		**IoT – IoT-23**		**IoT – TON_IoT**		**VPN – ISCX**		**VPN – VNAT**	
11,570	3	122,040	181	1,346	206	45,049	170	9,691	0
7	176	550	377,229	52	498,396	509	4,272	4	305

Table B.11
Confusion matrix for multiclass classification of Botnet using CTU-13 dataset.

		Predicted labels					
		Clear	**Donbot**	**Fast flux**	**Neris**	**Qvod**	**Rbot**
True labels	**Clear**	23,428	0	0	0	0	0
	Donbot	0	3	2	0	0	0
	Fast flux	2	0	566	17	0	0
	Neris	1	0	11	923	0	0
	Qvod	0	0	1	0	38	0
	Rbot	2	0	1	1	0	4

Table B.12
Confusion matrix for multiclass classification of VPN using ISCX dataset.

		Predicted labels					
		CHAT	EMAIL	FileTransfer	P2P	STREAMING	VOIP
True labels	CHAT	904	0	33	0	3	48
	EMAIL	6	79	1	1	1	7
	FileTransfer	24	0	295	6	13	33
	P2P	3	0	3	139	0	6
	STREAMING	9	0	17	7	255	17
	VOIP	29	0	24	2	5	3,030

Table B.13
Confusion matrix for multiclass classification of VPN using VNAT dataset.

		Predicted labels				
		C2	CHAT	FILE TRANSFER	STREAMING	VoIP
True labels	C2	3,906	2	3	16	0
	CHAT	9	918	3	6	0
	FILE TRANSFER	5	2	4,095	11	0
	STREAMING	8	2	2	862	0
	VoIP	2	0	0	0	148

Table B.14
Confusion matrix for multiclass classification of IoT Malware by using Edge-IIoTset dataset.

		Predicted labels											
		SQL injection	XSS	Backdoor	Clear	DDoS	MitM	OS fingerprinting	Password attack	Port scanning	Ransomware	Uploading attack	Vulnerability scanner
True labels	SQL injection	982	1	0	0	0	0	0	1	0	0	0	13
	XSS	0	354	0	0	1	0	0	0	0	0	0	0
	Backdoor	0	0	9	1	0	0	0	0	0	0	0	0
	Clear	0	0	0	229,064	2	0	0	0	0	0	0	0
	DDoS	0	0	0	2	2,358	0	0	1	0	0	0	2
	MitM	0	0	0	0	0	15	0	0	6	0	0	0
	OS fingerprinting	0	0	0	1	0	0	0	0	54	0	0	0
	Password attack	0	0	0	1	0	0	0	12,259	0	0	0	1
	Port scanning	0	0	0	1	1	0	0	0	3,551	0	0	0
	Ransomware	0	0	0	0	1	0	0	0	0	9	0	0
	Uploading attack	0	0	0	0	0	0	0	0	0	0	400	1
	Vulnerability scanner	13	0	0	0	0	0	0	2	0	0	0	893

Table B.15
Confusion matrix for multiclass classification of IoT Malware using TON_IoT dataset.

		Predicted labels								
		Backdoor	Clear	DoS	Injection	MitM	Password	Runsomware	Scanning	XSS
True labels	Backdoor	3,928	10	153	55	13	199	151	240	216
	Clear	7	719	14	9	1	6	3	31	10
	DoS	230	9	44,732	173	11	358	56	266	311
	Injection	63	3	268	27,794	4	400	25	132	629
	MitM	47	2	54	6	76	26	24	55	49
	Password	177	5	388	482	16	32,572	42	169	348
	Runsomware	194	6	97	33	12	106	953	96	137
	Scanning	450	10	317	121	19	204	51	21,867	778
	XSS	141	6	302	307	4	317	49	272	107,384

Table B.16
Confusion matrix for multiclass classification of Intrusion in IDS system using CIC-IDS-2017 dataset.

		Predicted labels								
		BENIGN	Bot	DDoS	FTP-Patator	Heartbleed	Infiltration	PortScan	SSH-Patator	Web Attack
True labels	BENIGN	485,953	92	381	0	0	0	5	0	0
	Bot	213	500	0	0	0	0	0	0	0
	DDoS	251	0	134,320	0	0	0	2	0	3
	FTP-Patator	8	0	0	2,277	0	0	0	0	0
	Heartbleed	1	0	0	0	2	0	0	0	0
	Infiltration	13	0	0	0	0	0	0	0	0
	PortScan	20	0	219	0	0	0	224	1	0
	SSH-Patator	22	0	0	0	0	0	1	1,684	0
	Web Attack	50	0	0	0	0	0	0	0	609

Table B.17
Confusion matrix for multiclass classification of Intrusion in IDS system using UNSW dataset.

		Predicted labels									
		Analysis	Backdoor	Clear	DoS	Exploits	Fuzzers	Generic	Reconnaissance	Shellcode	Worms
True labels	Analysis	0	0	89	1	11	0	0	0	0	0
	Backdoor	0	1,892	5	2	32	0	5	4	4	1
	Clear	0	14	563,592	13	341	1,352	41	141	31	0
	DoS	0	0	78	225	623	19	24	20	12	3
	Exploits	0	2	304	60	6234	35	105	156	36	13
	Fuzzers	0	0	2,358	4	41	3,228	3	3	1	0
	Generic	0	0	28	15	245	13	795	2	4	0
	Reconnaissance	0	84	19	2	281	5	2	2,907	0	0
	Shellcode	0	0	8	5	35	0	5	0	363	0
	Worms	0	0	1	0	21	0	1	0	0	23

Table B.18
Confusion matrix for multiclass classification of TOR.

		Predicted labels							
		AUDIO	BROWSING	CHAT	FILE TRANSFER	MAIL	P2P	VIDEO	VOIP
True labels	AUDIO	518	41	0	0	1	27	2	8
	BROWSING	23	8,038	2	1	1	99	29	2
	CHAT	4	26	63	1	0	6	7	7
	FILE TRANSFER	2	7	0	566	0	13	3	0
	MAIL	1	23	0	0	63	6	1	0
	P2P	0	71	0	3	0	9,542	0	3
	VIDEO	11	199	1	4	0	17	332	3
	VOIP	6	37	1	1	0	15	13	150

Appendix C. Description of merged multiclass dataset

The complex multiclass dataset was created from following datasets: 1. CTU-13 [53], 2. CESNET-MINER22 [57], 3. CIC-Bell-DNS [15], 4. DoH-Real-world [61], 5. CIC-DoH-Brw [59], 6. Bot-IoT [62], 7. HTTPS Brute-force [55], 8. CIC-IDS-2017 [43], 9. UNSW-NB-15 [70], 10. Edge-IIoTset [87], 11. IoT-23 [64], 12. TON_IoT [68], 13. ISCX-Tor-2016 [72], and 14. ISCX-VPN-2016 [73], and 15. VNAT [74].

In the created dataset, we merge classes with the same traffic resulting in the following 44 classes: Botnet-Neris, Clear, Botnet-RBot, Botnet-Fast_flux, Botnet-Donbot, Botnet-Sogou, Botnet-Qvod, Cryptomining, DNS-malware, DoH, Backdoor, DoS, MitM, OS_fingerprinting, Password_attack, Scanning, Ransomware, Injection, Uploading_attack, Vulnerability_scanner, XSS, HTTPS-Brute-force, Bot, PortScan, Infiltration, FTP-Patator, SSH-Patator, Heartbleed, AUDIO, BROWSING, CHAT, File Transfer, EMAIL, P2P, VIDEO, VOIP, Fuzzers, Exploits, Shellcode, Worms, Reconnaissance, Backdoor, Analysis, and C2.

Appendix D. Feature importance

The individual feature importances across the evaluated classification tasks are shown in Figs. D.10–D.12.

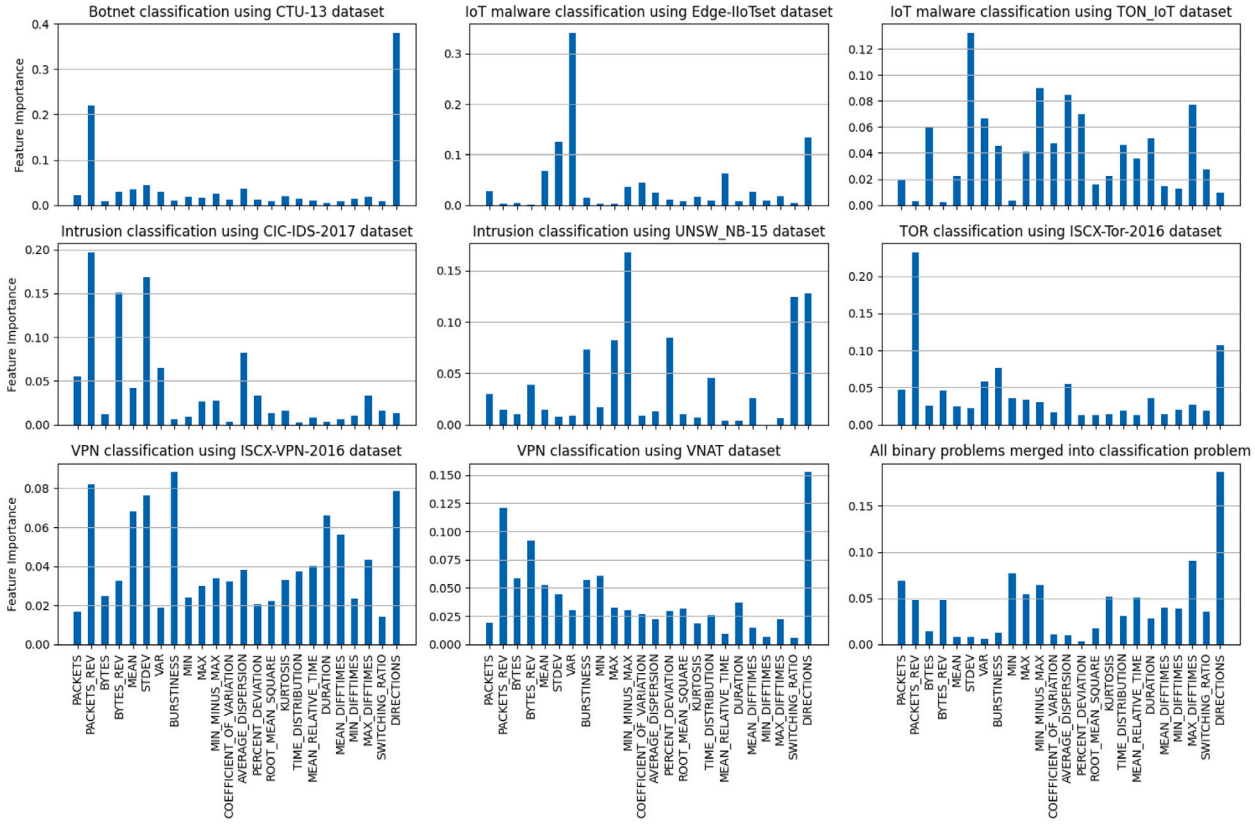


Fig. D.10. Feature importance for multiclass classification.

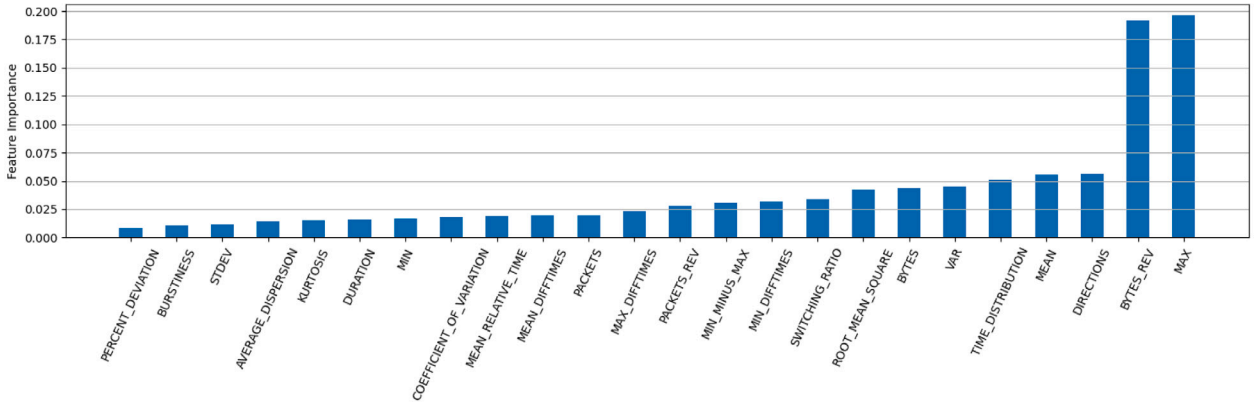


Fig. D.11. Feature importance for classification on all dataset merged into multiclass problem with 44 classes.

Fig. D.12. Feature importance for binary classification.

References

[1] A. Heidari, N.J. Navimipour, M. Unal, Applications of ML/DL in the management of smart cities and societies based on new trends in information technologies: A systematic literature review, Sustainable Cities Soc. 85 (2022) 104089, http://dx.doi.org/10.1016/j.scs.2022.104089.

[2] A.P. Plageras, K.E. Psannis, C. Stergiou, H. Wang, B. Gupta, Efficient IoT-based sensor BIG Data collection–processing and analysis in smart buildings, Future Gener. Comput. Syst. 82 (2018) 349–357, http://dx.doi.org/10.1016/j.future.2017.09.082.

[3] X. Pan, S. Yamaguchi, Machine learning white-hat worm launcher for tactical response by zoning in botnet defense system, Sensors 22 (13) (2022) 4666, http://dx.doi.org/10.3390/s22134666.

[4] E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.3, 2018, http://dx.doi.org/10.17487/RFC8446, Request for Comments 8446, RFC Editor, URL https://www.rfc-editor.org/info/rfc8446.

[5] P.E. Hoffman, P. McManus, DNS queries over HTTPS (DoH), 2018, http://dx.doi.org/10.17487/RFC8484, Request for Comments 8484 RFC Editor URL https://www.rfc-editor.org/info/rfc8484.

[6] E. Rescorla, et al., TLS Encrypted Client Hello, Internet-Draft draft-ietf-tls-esni-16, Internet Engineering Task Force, 2023.

[7] D. Paraskevi, J. Fajfer, N. Müller, E. Papadogiannaki, E. Rekleitis, F. Střasák, Encrypted Traffic Analysis, Use Cases & Security Challenges, European Union Angency for Cybersecurity, 2020, URL https://www.enisa.europa.eu/publications/encrypted-traffic-analysis.

[8] A. Aqil, K. Khalil, A.O. Atya, E.E. Papalexakis, S.V. Krishnamurthy, T. Jaeger, K.K. Ramakrishnan, P. Yu, A. Swami, Jaal: Towards network intrusion detection at ISP scale, in: Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 134–146, http://dx.doi.org/10.1145/3143361.3143399.

[9] B. Claise, B. Trammell, P. Aitken, Specification of the IP flow information export (IPFIX) protocol for the exchange of flow information, 2013, pp. 1–76, http://dx.doi.org/10.17487/RFC7011, RFC 7011.

[10] B. Claise, Cisco systems NetFlow services export version 9, 2004, pp. 1–33, http://dx.doi.org/10.17487/RFC3954, RFC 3954.

[11] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, A. Pras, Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX, IEEE Commun. Surv. Tutor. 16 (4) (2014) 2037–2064, http://dx.doi.org/10.1109/COMST.2014.2321898.

[12] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, B. Stiller, An overview of IP flow-based intrusion detection, IEEE Commun. Surv. Tutor. 12 (3) (2010) 343–356, http://dx.doi.org/10.1109/SURV.2010.032210.00054.

[13] K. Jerabek, K. Hynek, O. Rysavy, I. Burgetova, DNS over HTTPS detection using standard flow telemetry, IEEE Access 11 (2023) 50000–50012, http://dx.doi.org/10.1109/ACCESS.2023.3275744.

[14] T. Zebin, et al., An explainable AI-based intrusion detection system for DNS over HTTPS (DoH) attacks, IEEE Trans. Inf. Forensics Secur. 17 (2022) 2339–2349, http://dx.doi.org/10.1109/TIFS.2022.3183390.

[15] S. Mahdavifar, et al., Classifying malicious domains using DNS traffic analysis, in: DASC/PiCom/CBDCom/CyberSciTech 2021, IEEE, 2021, pp. 60–67, http://dx.doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech52372.2021.00024.

[16] J. Luxemburk, T. Cejka, Fine-grained TLS services classification with reject option, Comput. Netw. 220 (2023) 109467, http://dx.doi.org/10.1016/j.comnet.2022.109467.

[17] S. Shaikh, C. Rupa, G. Srivastava, T. Reddy Gadekallu, Botnet attack intrusion detection in IoT enabled automated guided vehicles, in: 2022 IEEE International Conference on Big Data, Big Data, IEEE, 2022, http://dx.doi.org/10.1109/bigdata55660.2022.10020355.

[18] Y. Madwanna, B. Annappa, H. Sneha, et al., YARS-IDS: A novel IDS for multi-class classification, in: 2023 IEEE 8th International Conference for Convergence in Technology, I2CT, IEEE, 2023, pp. 1–6.

[19] A. Heidari, N.J. Navimipour, M.A.J. Jamali, S. Akbarpour, A hybrid approach for latency and battery lifetime optimization in IoT devices through offloading and CNN learning, Sustain. Comput. Inform. Syst. 39 (2023) 100899, http://dx.doi.org/10.1016/j.suscom.2023.100899.

[20] A. Heidari, N.J. Navimipour, M.A.J. Jamali, S. Akbarpour, A green, secure, and deep intelligent method for dynamic IoT-edge-cloud offloading scenarios, Sustain. Comput. Inform. Syst. 38 (2023) 100859, http://dx.doi.org/10.1016/j.suscom.2023.100859.

[21] A. Heidari, M.A.J. Jamali, N.J. Navimipour, S. Akbarpour, A QoS-aware technique for computation offloading in IoT-edge platforms using a convolutional neural network and Markov decision process, IT Prof. 25 (1) (2023) 24–39, http://dx.doi.org/10.1109/mitp.2022.3217886.

[22] J. Luxemburk, K. Hynek, T. Čejka, Encrypted traffic classification: the QUIC case, in: 2023 7th Network Traffic Measurement and Analysis Conference, TMA, IEEE, 2023, pp. 1–10.

[23] Z. Tropková, et al., Novel HTTPS classifier driven by packet bursts, flows, and machine learning, in: CNSM 2021, IEEE, 2021, pp. 345–349, http://dx.doi.org/10.23919/CNSM52442.2021.9615561.

[24] R. Plný, et al., DeCrypto: Finding cryptocurrency miners on ISP networks, in: NordSec 2022, Vol. 13700, Springer, 2022, pp. 139–158, http://dx.doi.org/10.1007/978-3-031-22295-5_8.

[25] J. Velasco-Mata, V. González-Castro, E.F. Fernández, E. Alegre, Efficient detection of botnet traffic by features selection and decision trees, IEEE Access 9 (2021) 120567–120579.

[26] J. Koumar, K. Hynek, T. Čejka, Network traffic classification based on single flow time series analysis, in: 2023 19th International Conference on Network and Service Management, CNSM, 2023, pp. 1–7, http://dx.doi.org/10.23919/CNSM59352.2023.10327876.

[27] J. Koumar, T. Čejka, Unevenly spaced time series from network traffic, in: 2023 7th Network Traffic Measurement and Analysis Conference, TMA, IEEE, 2023, pp. 1–4.

[28] J. Koumar, K. Hynek, J. Pešek, T. Čejka, Network Traffic Datasets with Novel Extended IP Flow Called Nettisa Flow, Zenodo, 2023, http://dx.doi.org/10.5281/zenodo.8301043.

[29] P. Velan, M. Čermák, P. Čeleda, M. Drašar, A survey of methods for encrypted traffic classification and analysis, Int. J. Netw. Manage. 25 (5) (2015) 355–374.

[30] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, P.E. Hoffman, Specification for DNS over Transport Layer Security (TLS), 2016, http://dx.doi.org/10.17487/RFC7858, Request for Comments 7858, RFC Editor, URL https://www.rfc-editor.org/info/rfc7858.

[31] S. García, K. Hynek, D. Vekshin, T. Čejka, A. Wasicek, Large scale measurement on the adoption of encrypted DNS, 2021, arXiv preprint arXiv:2107.04436.

[32] Z. Tsiatsikas, G. Karopoulos, G. Kambourakis, Measuring the adoption of TLS encrypted client hello extension and its forebear in the wild, in: European Symposium on Research in Computer Security, Springer, 2022, pp. 177–190.

[33] D. Shamsimukhametov, A. Kurapov, M. Liubogoshchev, E. Khorov, Is encrypted ClientHello a challenge for traffic classification? IEEE Access 10 (2022) 77883–77897.

[34] G. Aceto, et al., DISTILLER: encrypted traffic classification via multimodal multitask deep learning, J. Netw. Comput. Appl. 183–184 (2021) 102985, http://dx.doi.org/10.1016/j.jnca.2021.102985.

[35] M. Lopez-Martin, et al., Network traffic classifier with convolutional and recurrent neural networks for internet of things, IEEE Access 5 (2017) 18042–18050.

[36] D. Vekshin, K. Hynek, T. Cejka, Doh insight: Detecting DNS over HTTPS by machine learning, in: M. Volkamer, C. Wressnegger (Eds.), ARES 2020: The 15th International Conference on Availability, Reliability and Security, Virtual Event, Ireland, August 25-28, 2020, ACM, 2020, pp. 87:1–87:8, http://dx.doi.org/10.1145/3407023.3409192.

[37] K. Hynek, T. Beneš, T. Čejka, H. 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, Springer, 2020, pp. 49–63.

[38] Z. Amiri, A. Heidari, N.J. Navimipour, M. Unal, A. Mousavi, Adventures in data analysis: a systematic review of deep learning techniques for pattern recognition in cyber-physical-social systems, Multimedia Tools Appl. (2023) http://dx.doi.org/10.1007/s11042-023-16382-x.

[39] Z. Chen, K. He, J. Li, Y. 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, IEEE, 2017, pp. 1271–1276.

[40] T. Shapira, et al., Flowpic: Encrypted internet traffic classification is as easy as image recognition, in: IEEE INFOCOM 2019, IEEE, 2019.

[41] R. Hofstede, M. Jonker, A. Sperotto, A. Pras, Flow-based web application brute-force attack and compromise detection, J. Netw. Syst. Manage. (2017) http://dx.doi.org/10.1007/s10922-017-9421-4.

[42] W. Wang, M. Zhu, J. Wang, X. Zeng, Z. Yang, End-to-end encrypted traffic classification with one-dimensional convolution neural networks, in: 2017 IEEE International Conference on Intelligence and Security Informatics, ISI, IEEE, 2017, pp. 43–48.

[43] I. Sharafaldin, et al., Toward generating a new intrusion detection dataset and intrusion traffic characterization, in: ICISSp, Vol. 1, 2018, pp. 108–116.

[44] G. Agrafiotis, 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, 2022, pp. 1–7.

[45] H. Ding, et al., Imbalanced data classification: A KNN and generative adversarial networks-based hybrid approach for intrusion detection, Future Gener. Comput. Syst. 131 (2022) 240–254.

[46] I. Cvitić, D. Peraković, M. Periša, B. Gupta, Ensemble machine learning approach for classification of IoT devices in smart home, Int. J. Mach. Learn. Cybern. 12 (11) (2021) 3179–3202, http://dx.doi.org/10.1007/s13042-020-01241-0.

[47] M. MontazeriShatoori, et al., Detection of DoH tunnels using time-series classification of encrypted traffic, in: DASC/PiCom/CBDCom/CyberSciTech 2020, IEEE, 2020, pp. 63–70, http://dx.doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00026.

[48] A. Moore, D. Zuev, C. 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.

[49] M. Nasr, A. Houmansadr, A. 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, Association for Computing Machinery, New York, NY, USA, 2017, pp. 2053–2069, http://dx.doi.org/10.1145/3133956.3134074.

[50] S. Yeom, C. Choi, K. Kim, AutoEncoder Based Feature Extraction for Multi-Malicious Traffic Classification, in: SMA 2020, Association for Computing Machinery, New York, NY, USA, 2021, pp. 285–287, http://dx.doi.org/10.1145/3426020.3426093.

[51] R. Yan, R. Liu, Principal component analysis based network traffic classification, J. Comput. 9 (5) (2014) 1234–1240, http://dx.doi.org/10.4304/jcp.9.5.1234-1240.

[52] D. Uhříček, K. Hynek, T. Čejka, D. Kolář, BOTA: Explainable IoT malware detection in large networks, IEEE Internet Things J. 10 (10) (2023) 8416–8431, http://dx.doi.org/10.1109/JIOT.2022.3228816.

[53] S. García, et al., An empirical comparison of botnet detection methods, Comput. Secur. 45 (2014) 100–123, http://dx.doi.org/10.1016/j.cose.2014.05.011.

[54] G. Stergiopoulos, et al., Automatic detection of various malicious traffic using side channel features on TCP packets, in: ESORICS 2018, Vol. 11098, Springer, 2018, pp. 346–362, http://dx.doi.org/10.1007/978-3-319-99073-6_17.

[55] J. Luxemburk, K. Hynek, T. Cejka, HTTPS Brute-Force Dataset with Extended Network Flows, Zenodo, 2020, http://dx.doi.org/10.5281/zenodo.4275775.

[56] J. Luxemburk, et al., Detection of HTTPS brute-force attacks with packet-level feature set, in: CCWC 2021, 2021, pp. 0114–0122, http://dx.doi.org/10.1109/CCWC51732.2021.9375998.

[57] R. Plný, et al., Datasets of Cryptomining Communication, Zenodo, 2022, http://dx.doi.org/10.5281/zenodo.7189293.

[58] M. Kumaar, et al., A hybrid framework for intrusion detection in healthcare systems using deep learning, Front. Public Health 9 (2021).

[59] M. MontazeriShatoori, et al., Detection of doh tunnels using time-series classification of encrypted traffic, in: DASC/PiCom/CBDCom/CyberSciTech 2020, IEEE, 2020, pp. 63–70.

[60] M. Behnke, N. Briner, D. Cullen, K. Schwerdtfeger, J. Warren, R. Basnet, T. Doleck, Feature engineering and machine learning model comparison for malicious activity detection in the dns-over-https protocol, IEEE Access 9 (2021) 129902–129916.

[61] K. Jeřábek, et al., Collection of datasets with DNS over HTTPS traffic, Data Brief 42 (2022) 108310, http://dx.doi.org/10.1016/j.dib.2022.108310.

[62] N. 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 (2019) 779–796, http://dx.doi.org/10.1016/j.future.2019.05.041.

[63] M. 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 (2020) 433–442, http://dx.doi.org/10.1016/j.future.2020.02.017.

[64] S. Garcia, et al., IoT-23: A Labeled Dataset with Malicious and Benign IoT Network Traffic, Zenodo, 2020, http://dx.doi.org/10.5281/zenodo.4743746, More details here https://www.stratosphereips.org/datasets-iot23.

[65] A.K. Sahu, et al., Internet of Things attack detection using hybrid deep learning model, Comput. Commun. 176 (2021) 146–154.

[66] M.A. Ferrag, et al., Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications: Centralized and Federated Learning, IEEE Dataport, 2022, http://dx.doi.org/10.21227/mbc1-1h68.

[67] A. Khacha, R. Saadouni, Y. Harbi, Z. Aliouat, Hybrid deep learning-based intrusion detection system for industrial internet of things, in: 2022 5th International Symposium on Informatics and Its Applications, ISIA, IEEE, 2022, pp. 1–6.

[68] N. Moustafa, A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets, Sustainable Cities Soc. 72 (2021) 102994.

[69] J. Dai, X. Xu, F. Xiao, GLADS: A global-local attention data selection model for multimodal multitask encrypted traffic classification of IoT, Comput. Netw. 225 (2023) 109652.

[70] N. Moustafa, J. 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, IEEE, 2015, pp. 1–6.

[71] M. Nawir, A. Amir, O.B. Lynn, N. Yaakob, R. Badlishah Ahmad, Performances of machine learning algorithms for binary classification of network anomaly detection system, J. Phys. Conf. Ser. 1018 (2018) 012015.

[72] A.H. Lashkari, et al., Characterization of tor traffic using time based features, in: ICISSP 2017, SciTePress, 2017, pp. 253–262, http://dx.doi.org/10.5220/0006105602530262.

[73] G. Draper-Gil, et al., Characterization of encrypted and VPN traffic using time-related, in: Proceedings of the 2nd International Conference on Information Systems Security and Privacy, ICISSP, 2016, pp. 407–414.

[74] S. Jorgensen, et al., Extensible machine learning for encrypted network traffic application labeling via uncertainty quantification, 2022, http://dx.doi.org/10.48550/arXiv.2205.05628, CoRR abs/2205.05628, arXiv:2205.05628.

[75] G. Marín, et al., Deep in the dark - deep learning-based malware traffic detection without expert knowledge, in: SPW 2019, 2019, pp. 36–42, http://dx.doi.org/10.1109/SPW.2019.00019.

[76] I. Tareq, B.M. Elbagoury, S. El-Regaily, E.-S.M. El-Horbaty, Analysis of ToN-IoT, UNW-NB15, and edge-IIoT datasets using DL in cybersecurity for IoT, Appl. Sci. 12 (19) (2022) 9572.

[77] Y.N. Kunang, S. Nurmaini, D. Stiawan, B.Y. Suprapto, Attack classification of an intrusion detection system using deep learning and hyperparameter optimization, J. Inf. Secur. Appl. 58 (2021) 102804.

[78] M. Dener, S. Al, G. Ok, RFSE-GRU: Data balanced classification model for mobile encrypted traffic in big data environment, IEEE Access 11 (2023) 21831–21847.

[79] N. Brownlee, K.C. Claffy, Understanding internet traffic streams: Dragonflies and tortoises, IEEE Commun. Mag. 40 (10) (2002) 110–117.

[80] J. Luxemburk, et al., CESNET-QUIC22: a large one-month QUIC network traffic dataset from backbone lines, Data Brief (2023) 108888.

[81] J.D. Hamilton, Time Series Analysis, Princeton University Press, 2020.

[82] H.Z. Moayedi, M. Masnadi-Shirazi, Arima model for network traffic prediction and anomaly detection, in: 2008 International Symposium on Information Technology, Vol. 4, IEEE, 2008, pp. 1–6.

[83] A.A. Cook, G. Mısırlı, Z. Fan, Anomaly detection for IoT time-series data: A survey, IEEE Internet Things J. 7 (7) (2019) 6481–6494.

[84] B. Ghojogh, M. Crowley, The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial, 2019, arXiv preprint arXiv:1905.12787.

[85] J. Bergstra, D. Yamins, D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, in: International Conference on Machine Learning, PMLR, 2013, pp. 115–123.

[86] Linux Foundation, Data plane development kit (DPDK), 2015, URL http://www.dpdk.org.

[87] M.A. Ferrag, et al., Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning, IEEE Access 10 (2022) 40281–40306, http://dx.doi.org/10.1109/ACCESS.2022.3165809.

Josef Koumar is currently pursuing a Ph.D. with the Faculty of Information Technology, Czech Technical University in Prague, focusing on detecting threats and classifications of encrypted network traffic using time series analysis and also interested in anomaly detection. He is also working as a Researcher at CESNET a.l.e., the operator of the Czech national research and educational networks.

Karel Hynek is a Ph.D. candidate at the Faculty of Information Technology, Czech Technical University in Prague. He started his Ph.D. study in 2019 with a topic related to network security and network monitoring. Since his bachelor's study, Ing. Hynek has worked on several national and international research projects related to security, and he is currently a key member of the network traffic monitoring research team at the faculty.

Jaroslav Pešek is a Ph.D. student at the Faculty of Information Technology, Czech Technical University in Prague. He started in 2023 with focus on an effective and scalable machine learning pipeline for network traffic monitoring and security. Jaroslav Pesek is also involved in network traffic measurement and statistical processing and he is a key member of the network traffic monitoring research team at CESNET a.l.e.

Tomáš Čejka is an assistant professor at FIT CTU in Prague. He teaches network security course, and supervises a research group of bachelor/master/Ph.D. students. Meanwhile, he also works at CESNET a.l.e., the operator of the Czech national research and education network, as a researcher and team leader.