IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 30 October 2023, accepted 21 December 2023, date of publication 29 December 2023,
date of current version 8 January 2024.

Digital Object Identifier 10.1109/ACCESS.2023.3348234

RESEARCH ARTICLE

Enhancing Detection of Malicious Traffic Through FPGA-Based Frequency Transformation and Machine Learning

ZHENGUO HU[1], HIROKAZU HASEGAWA[2], (Member, IEEE), YUKIKO YAMAGUCHI[3], AND HAJIME SHIMADA[3], (Member, IEEE)

[1]Graduate School of Informatics, Nagoya University, Nagoya 464-8601, Japan
[2]Center for Strategic Cyber Resilience Research and Development, National Institute of Informatics, Tokyo 101-8430, Japan
[3]Information Technology Center, Nagoya University, Nagoya 464-8601, Japan

Corresponding author: Zhenguo Hu (tu36rz9u@gmail.com)

This work was supported in part by the Nagoya University Interdisciplinary Frontier Fellowship funded by Nagoya University and Japan Science and Technology Agency (JST) under Grant JPMJFS2120, and in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP23H03396 and Grant JP19K11961.

ABSTRACT In recent years, with the development of modern network, in order to avoid threats caused by cyber attacks, it is important to understand how to implement effective security measures. Malicious traffic detection is an advanced technique, it employs several approaches to distinguish traffic whether it is benign or malicious. Traditional malicious traffic detection methods are usually based on pre-defined signatures. However, limited by the size and timeliness of the signature library, they are usually unable to detect unknown cyber attacks such as zero-day attacks or new malware variants. In order to solve this problem, we propose a machine learning based method to detect realtime malicious traffic. It is divided into two parts: feature extraction on FPGA and abnormal traffic detection on Linux host machine. On the feature extraction part, instead of using conventional network traffic features, we propose a frequency transformation based feature extraction method to extract frequency domain features from network traffic. At the same time, in order to improve the speed of feature extraction and reduce CPU resource usage, we implement the processes required for feature extraction inside the FPGA board. On the abnormal traffic detection part, we use AF-Packet and ring buffer to capture the features, and load a pre-trained model into the CatBoost framework in advance to execute inference process. We evaluate our proposed system on a Xilinx Alveo U50 accelerator card and a Linux host machine. The evaluation results show that we achieve about 0.98 detection accuracy with low resource usage and good realtime detection throughput.

INDEX TERMS FPGA, malicious traffic detection, machine learning.

I. INTRODUCTION

Nowadays, with the development of network facilities, the Internet has entered many aspects of people's lives. While the Internet advancement has brought benefits, it has also given rise to cyber threats, that pose a persistent challenge to the security of information network constructions. Cyber attackers try to exploit vulnerabilities, compromise data and even disrupt critical services. Recent reports indicate that cyber threats are becoming more and more widespread and complex [1], [2], [3]. One of the manifestations of cyber threats is malicious traffic. Malicious traffic refers to malicious or unauthorized traffic such as Distributed Denial of Service (DDoS) attacks, malware distribution and so on. Since attackers develop sophisticated techniques to evade detection, it is a constant challenge for cybersecurity researchers. In order to protect network environment and prevent damages caused by similar cyber threats, the detection and inspection of malicious traffic are becoming important for ensuring security and integrity of network environments.

The associate editor coordinating the review of this manuscript and approving it for publication was Xiong Luo.

TABLE 1. Analysis between our design with other detection methods.

Method Type	Research	Realtime	Unknown Threat	FPGA Offload
Traditional Method	Suricata	✓	×	×
	K. Jack [27]	✓	×	✓
ML based Method	A. Javaid [31]	×	✓	×
	Our Design	✓	✓	✓

Malicious traffic detection involves the use of advanced approaches to identify suspicious network behaviors, analyze traffic and distinguish it from multiple methods. Meanwhile, Intrusion Detection System (IDS) or Network Intrusion Detection System (NIDS) widely uses this technology among the existing protection approaches responsible for detecting malicious activities [4]. Traditional malicious traffic detection methods are usually based on pre-defined signatures or patterns, they are used to identify known malicious traffic activities in network environments [5], [6]. As traffic enters the detection system, it is continuously inspected and compared against the defined signatures. When the signature matches and hits, the traffic is marked as potentially malicious for further analysis or defense actions. The representative systems in signature based malicious traffic detection are Snort[1] and Suricata.[2] They are widely used in many institutions and organizations. Although these methods have great recognition rate, but they rely on the pre-defined signatures or patterns of known cyber threats. It is usually incapable of detecting unknown or zero-day attacks [7], [8], [9], because there are no corresponding signatures for such threats. Therefore, more and more researchers are turning to the study of machine learning based malicious traffic detection.

Unlike the traditional malicious traffic detection methods, machine learning (ML) based methods leverage the advantages of artificial intelligence and statistical models to identify the malicious traffic. They overcome the limitations of traditional malicious traffic detection methods that rely on manually selected signatures or patterns, and can effectively detect new malicious traffic. By extracting traffic features (e.g., port number and packet length) and learning from historical traffic data, these methods have the ability to recognize both known and unknown malicious traffic, which increases the defense abilities against new cyber threats. There are a lot of researches involving machine learning models to detect malicious traffic [10], [11], [12], [13]. Although machine learning based methods have many advantages over traditional signature or pattern based detection methods. However, most of them are dedicated to detect offline malicious traffic files [14], [15], [16], [17], which will cause delays in responding to malicious behaviors or cyber attacks. In that case, in order to overcome this limitation and adapt to the rapidly developing network environment, we urgently need a method to detect malicious traffic in realtime.

In this paper, we propose a malicious traffic detection method which utilizes the FPGA-based frequency transformation and machine learning to detect realtime traffic. Table 1 shows the analysis between our design with other detection methods. Our research motivation is driven by detecting increasing attacks, and providing effective realtime protection whether on the Internet and Intranet. Unlike the task of traffic classification [18], [19], [20], our system does not categorize traffic and identify the type of services. We focus on detecting whether the traffic flowing through the system is benign or malicious. Instead of relying on the CPU computation heavily, we offload most of the CPU-bound tasks to the FPGA board to reduce the CPU usage as much as possible. On the one hand, our system combines the unique hardware advantages of FPGA and utilizes hardware circuits to process the incoming traffic such as traffic decoding and feature extraction, which reduces detection latency while improving system performance. On the other hand, we rely on the fast packet capturing of AF-Packet and ring buffer, as well as the inference advantages of CatBoost, to achieve rapid and realtime detection of abnormal traffic.

The realtime traffic to be detected first inputs into our proposed detection system through the QSFP28 port, we adapt an FPGA board to receive and decode the original data packets directly. In order to execute frequency transformation rapidly, we build a four-way parallel real transformation of Fast Fourier Transform (FFT) to extract the frequency domain features. In addition, we adapt the CatBoost framework for the training of malicious traffic detection model and use it to execute rapid inference process.

The contributions of our work are shown as follows:

- We present a malicious traffic detection method which utilizes the FPGA-based frequency transformation to detect realtime malicious traffic.
- We execute the operations of traffic receiving and decoding, frequency domain features extraction and transportation inside the FPGA board.
- We train an abnormal traffic detection model and import it into the CatBoost framework to achieve fast inference.
- We design an experiment to evaluate the proposed system, which shows it has high accuracy and good realtime detection throughput.

The rest of this paper is organized as follows. In Section II, we review the related researches about malicious traffic detection and prevention. In Section III, we introduce the architecture of our proposed system, and show the detail of each design part. We also conduct an experiment and evaluate the prototype system in Section IV. In Section V, we discuss the current limitations and show the advantages of our proposed method. Finally, we conclude this paper and discuss the future work in Section VI.

[1]https://www.snort.org/
[2]https://suricata.io/

II. RELATED WORK

Over the last few years, with the rapid expansion of network infrastructures, the demand for detecting malicious traffic is gradually increasing. Many researchers have made a lot of efforts to find effective solutions on how to detect malicious traffic. In this section, we mainly review the development of malicious traffic detection technologies, including traditional approaches and machine learning based approaches, as well as related work on malicious traffic prevention.

A. TRADITIONAL MALICIOUS TRAFFIC DETECTION

Traditional malicious traffic detection methods typically use pre-defined signatures or patterns to search for known threats. It means when a signature matches with a previous attack that already exists in the database, the system will trigger a detection alarm [22]. The pre-defined signatures or patterns are generated by many cybersecurity experts or proprietary vendors [23]. In the paper [5], Borders et al. present Chimera to help improve intrusion detection. It is a declarative query language for network traffic processing. In this research [6], Jamshed et al. propose a highly-scalable software-based detection approach called Kargus, it exploits the potential of commodity computing hardware to detect malicious traffic. By using the high processing parallelism with CPUs and GPUs, it provides the processing function of multiple packets for every IDS function call. Nam et al. present a Suricata-based method on Many-Core Processors (MCPs) called Haetae [24]. It uses high parallelism to analyze the traffic flows. In addition, it can offload some packet processing tasks to the programmable network interface cards. When it meets high load tasks, Haetae can also offload traffic to the CPU side. In order to virtualize NIDS and scale to process traffic variations, Li et al. propose a NIDS architecture called vNIDS [25]. It minimizes the sharing overhead in virtualized environments and employs detection state sharing to achieve detection in virtualizing NIDSes. At the same time, it also uses program slicing to solve the problem of non-monolithic NIDS provisioning.

In addition to software-based solutions, many researchers have also utilized proprietary hardware to achieve signature based malicious traffic detection. Cascarano et al. utilize the parallel computing characteristics of GPU to propose a regular expression pattern matching engine [26]. It uses Nondeterministic Finite Automaton (NFA) to match data packets and can handle multiple of them simultaneously. In the paper [27], Jaic et al. present SFAOENIDS, which is a hybrid NIDS that uses an FPGA integrated with a network adapter to achieve pattern matching processing. In addition, Zhao et al. propose an intrusion detection or prevention method called Pigasus [28]. Pigasus adapts FPGA to deal with large malicious traffic detection. It offloads the majority of network packets processing and control flows to ensure most packets are processed inside the FPGA board, and combines with a revised Snort to achieve the traffic detection process. Both software and hardware based methods have been implemented to detect known cyber threats.

B. ML BASED MALICIOUS TRAFFIC DETECTION

As a promising method, machine learning based malicious traffic detection is gradually developing in recent years. In this research [29], Salama et al. propose a hybrid detection method which combines deep belief network and support vector machine to achieve intrusion detection. They use the NSL-KDD dataset to evaluate their system and the evaluation results indicate that their system has a high detection accuracy. Fiore et al. develop a detection system that use the Discriminative Restricted Boltzmann Machine to achieve network anomaly detection [30]. Invernizzi et al. present an infection detection system that is called Nazca [16]. Through studying behaviors of the clients download and install malware in real-world network environments, it can detect malware that is previously-unseen with low false positive rates. In order to detect various cyber attacks such as zero-day attacks or Advanced Persistent Threats (APTs), Bartos et al. introduce a system which the goal is to detect both known and previously-unseen cyber threats [17]. It adapts statistical feature representation and learns to recognize malicious behaviors to detect security threats. Javaid et al. present a NIDS to help system administrators to detect security breaches [31]. They employ Self-taught Learning (STL) of a NIDS to detect network intrusion and evaluate their system on the NSL-KDD dataset. In this paper [21], Fu et al. propose to use frequency domain features and statistical clustering algorithm to detect malicious traffic. They adapt Intel Data Plane Development Kit (DPDK) to receive traffic and k-means to achieve clustering. In order to reduce CPU usage, we adapt FPGA to offload traffic processing tasks instead of using software, which helps us achieve a balance between performance and resources.

In addition, Nelms et al. propose an incident investigation system called WebWitness [32]. Their system has the ability to trace the chain of events such as malware downloads and uses these download paths to develop effective defenses. In the paper [33], Antón et al. show the performance of some ML detection approaches such as k-means clustering and random forest on an industrial scenario. Mirsky et al. present a neural network based NIDS named Kitsune [34]. It employs an ensemble of neural networks named autoencoders to differentiate between benign and abnormal traffic patterns. It has the ability to track the behavior of all network channels, and detect attacks with a good performance even on a Raspberry PI. In this paper [35], Manna and Alkasassbeh propose a ML based method that uses the REP Tree, J48 and random forest classifiers to detect the anomalies devices. The experiment results show the REP tree classifier has the highest accuracy. Kumari et al. present to use k-means clustering technique to detect intrusions [36]. Their research method can extract features from flow instances. In this research [37], Choi et al. develop a NIDS to achieve anomaly

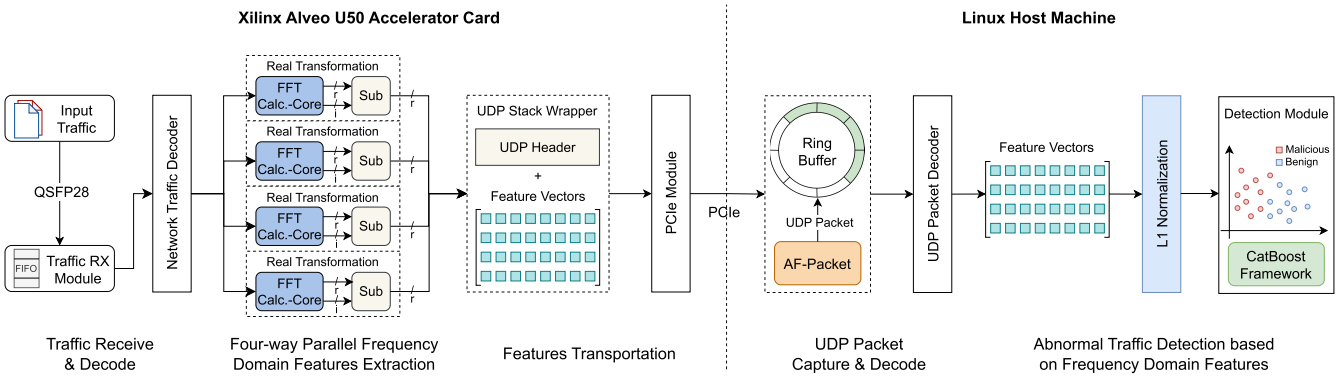


FIGURE 1. Overview of system architecture.

detection. By adapting an unsupervised learning algorithm autoencoder and setting a reconstruction loss threshold, their approach reaches a great performance accuracy.

C. MALICIOUS TRAFFIC PREVENTION

In addition to detect malicious traffic, many researchers also focus on the studies about how to prevent and block malicious traffic effectively. Xing et al. introduce an OpenFlow and Snort based intrusion prevention system which is named as SnortFlow [38]. This system enables many network reconfiguration actions that can not only achieve intrusion detection, but also has the ability to deploy corresponding countermeasures. In addition, in order to prevent and defense DDoS attacks, Fayaz et al. design a flexible and elastic DDoS defense system which is called Bohatei [39]. It brings flexibility and elasticity to the DDoS defense. Their defense system has the ability to outperform naive SDN implementations and enable resilient defenses. In this paper [40], Zheng et al. present a defense method which is named as RADAR. Their proposed system uses adaptive correlation analysis to detect and throttle DDoS attacks in realtime. It is also a practical system that can defend against a lot of flooding-based DDoS attacks. Scholz et al. develop programmable data plane prototypes in order to defense SYN floods attack [41]. The experiment results show that their proposed system can mitigate SYN floods with 10 GbE speed. It has the ability to achieve low end-to-end connection latencies and can be easily ported to other platforms.

III. MALICIOUS TRAFFIC DETECTION BASED ON FREQUENCY TRANSFORMATION

In this section, we propose a method which adapts frequency transformation to execute malicious traffic detection. The design of our proposed system is divided into two parts: frequency domain features extraction on the FPGA board and abnormal traffic detection on the Linux host machine. Fig. 1 shows the system architecture of our proposed method. We leverage the high performance of hardware to offload the frequency domain features extraction process to the FPGA board, and use the machine leaning algorithm to execute high efficient abnormal traffic detection.

A. FREQUENCY DOMAIN FEATURES EXTRACTION ON FPGA

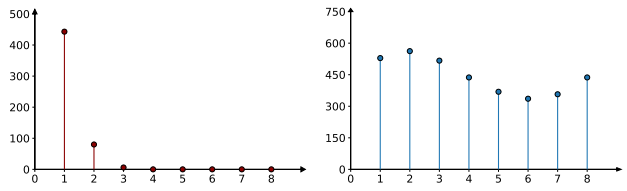
In order to extract features from network traffic, we need to decode and calculate the corresponding frequency information. The decoding processing and feature extraction are CPU-bound tasks that will cost many CPU resources. In order to reduce the resource usage and handle realtime traffic efficiently, we use a Xilinx Alveo U50 accelerator card[3] to implement the design including receiving and decoding traffic, extracting frequency domain features, and sending them to the host machine. The left part of Fig. 1 indicates the architecture of the FPGA design. We build our prototype system based on AMD OpenNIC project.[4]

1) ANALYSIS OF FREQUENCY DOMAIN FEATURES

During the malicious traffic detection process, traditional detection approaches usually adapt original traffic features to identity benign and malicious traffic. Different from the original traffic features, frequency domain features provide rich frequency domain information (e.g., amplitude and phase), that can help us to capture the diversity of the network traffic. Frequency transformation converts the signal from the time domain to the frequency domain. The information of the signal is presented in the form of different frequency components, which helps researchers to better analyze and process the signal. Expanding to the field of malicious traffic detection, the changes in the original traffic features will cause changes in the frequency domain components between benign and malicious traffic.

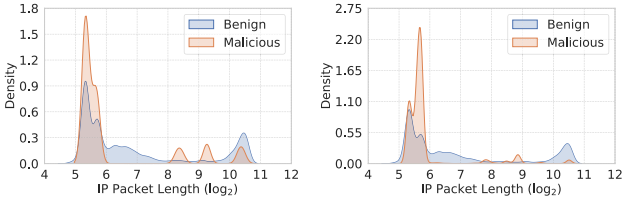
In the Fig. 2, we display the frequency transformation process of network traffic features. We can find after the frequency transformation, the frequency domain features present richer manifestation, that contributes to improve the ability of identifying different traffic samples. The extraction

[3]https://www.xilinx.com/products/boards-and-kits/alveo/u50.html
[4]https://github.com/Xilinx/open-nic



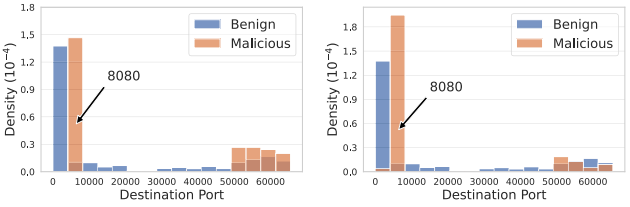
(a) Original features. (b) Frequency domain features.

FIGURE 2. Frequency transformation of network traffic features.



(a) Density distribution in Cridex. (b) Density distribution in Geodo.

FIGURE 3. Density distribution of IP packet length.



(a) Port distribution (Cridex). (b) Port distribution (Geodo).

FIGURE 4. Density distribution of destination port.

of frequency domain features can help researchers to achieve abnormal traffic detection by capturing and amplifying the changes among the traffic [42]. Although using frequency domain features can bring many benefits, but frequency transformation is usually a computationally intensive task and will consume many CPU resources. To solve this problem and ensure the realtime detection performance, we adapt FPGA to offload the time-consuming processes including frequency domain features extraction, which helps us to realize more efficient malicious traffic detection.

2) TRAFFIC RECEIVE AND DECODE

From the Fig. 1 we can see, the traffic RX module will receive the input traffic through the QSFP28 port. Here we adapt the Xilinx CMAC IP core as the receiver implementation. We also utilize a data FIFO inside the FPGA to cache unprocessed traffic. This peer-to-peer connection between input traffic and FPGA board can avoid overhead of network forwarding caused by the network interface card (NIC). After that, the network traffic decoder will decode every packet to extract original features. In our design, in order to improve universality of the system, we adapt three packet features including protocol, IP packet length and destination port. These features will be processed (e.g., encoding) and used for the next frequency domain features calculation.

In the network communication process, since protocol is usually used to distinguish various service types, there is a difference between benign and malicious traffic, so that we choose it as one of the original features and encode with protocol number. Besides, IP packet length and destination port are also two important original features. We process and analyze their density distribution separately to show the difference between benign and malicious samples. Fig. 3 indicates the density distribution of IP packet length in the experimental datasets [43], [44]. From the figure we can observe, benign traffic is distributed across different packet length ranges, while most of malicious traffic is mainly concentrated on small-sized packet areas. Fig. 4 shows the density distribution of destination port. We can find that in the experimental datasets, the distribution of malicious traffic is mainly concentrated on the port 8080. This is because Cridex and Geodo malware communicate with their servers through port 8080 to perform malicious behaviors such as uploading sensitive information. The special distribution patterns exhibited in the above features can help to distinguish between benign and malicious traffic effectively.

3) FOUR-WAY PARALLEL FREQUENCY DOMAIN FEATURES EXTRACTION

After obtaining the decoded features, we need to transform them to the frequency domain features. In our design, we adapt Fast Fourier Transform (FFT) approach as the frequency domain features extraction method. Fourier Transform (FT) is a kind of integral transformation, it is used to transform the original signals in time domain to frequency domain representation. Discrete Fourier Transform (DFT) is a discrete form of FT in both time and frequency domains. For a sequence of N samples like Equation 1 shows, the DFT is defined in Equation 2.

$$f(0),\ f(1),\ f(2),\ \dots\ f(k),\ \dots\ f(N-1) \tag{1}$$

$$DFT(n) = \sum_{k=0}^{N-1} f(k) e^{(-2\pi j/N)n \cdot k} \quad 0 \le n \le N-1 \tag{2}$$

The FFT is a mathematical method to compute the DFT rapidly. It reduces the time complexity from $O(N^2)$ to $O(N \log N)$. In our design, we adapt the radix-2 based FFT which is a form of Cooley-Tukey FFT to implement the feature extraction. As shown in the Fig. 1, the output frequency domain features from the FFT calculation cores are composed of complex numbers. Since machine learning algorithms usually accept features in the range of real numbers, the complex features can not be input into the model directly. In that case, we need to transform the complex features to real features. We adapt the subtractor implementation in the FPGA to achieve this goal inspired by Discrete Hartley

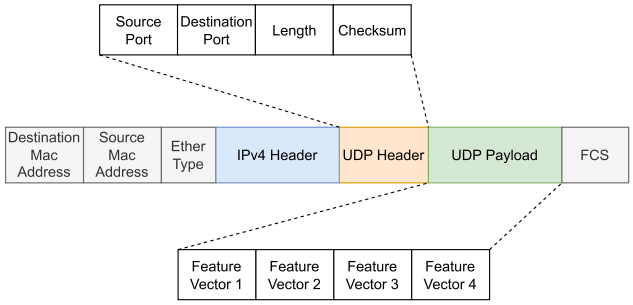


FIGURE 5. Structure of the packed UDP packet.

Transform (DHT). By subtracting the real and imaginary parts of the complex features, we obtain the frequency domain feature vectors composed of real numbers.

In the hardware design, in order to increase the calculation throughput, we implement a four-way parallel frequency domain features extraction architecture. By leveraging the parallelism of FPGA, this design can calculate the frequency domain feature vectors for four different sets of decoded features simultaneously. At the same time, in order to improve data transmission efficiency and reduce the transmission overhead of PCIe as much as possible, we encapsulate the four frequency feature vectors into one vector for the subsequent features transportation.

4) FEATURES TRANSPORTATION

The extracted frequency domain feature vectors need to be sent to the Linux host machine for abnormal traffic detection. Currently, speed of TCP is limited due to its packet transport guarantee mechanism, so that some advanced protocols (QUIC and HTTP/3) move to UDP with implementing packet transport guarantee in application layer. For example, HTTP/3 is an implementation of HTTP over QUIC. In our case, the transportation of feature vectors is not a stateful communication process. In order to maximize transmission bandwidth, we develop a simplified UDP stack to execute the transportation process. UDP stack adapts a connectionless communication method and runs on the layer 4 of the Open Systems Interconnection (OSI) model. It relies on the UDP header (e.g., port number and length) to mark the connection. In order to transport the feature vector, we build a packed UDP packet which is shown like Fig. 5.

From the figure we can see, the UDP header includes source port number, destination port number, length and checksum. The extracted four-way frequency domain feature vectors are packaged into one single block as the UDP payload. We add other needed information such as IPv4 header to build a complete UDP packet. At the same time, since the Xilinx Alveo U50 accelerator card uses PCIe Gen3 $\times$ 16 slot to communicate with the host machine, we design a PCIe module which uses the Xilinx QDMA IP core to transport the UDP data packets from FPGA board to the Linux host machine. By adapting this transportation approach, FPGA can efficiently transmit the extracted feature vectors to the software stack in realtime.

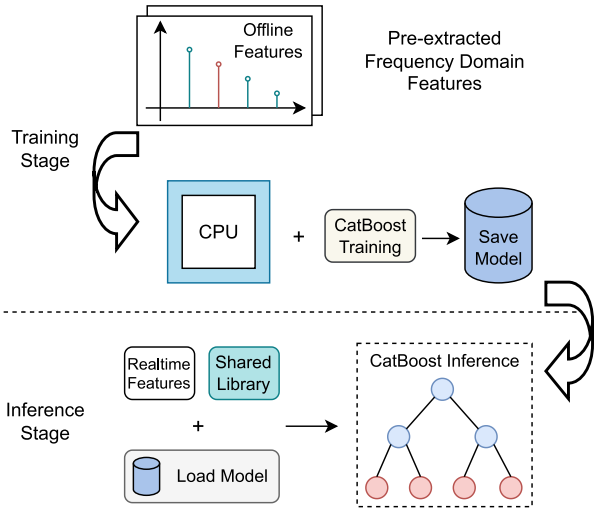


FIGURE 6. Overview of training stage and inference stage with CatBoost framework.

B. ABNORMAL TRAFFIC DETECTION ON LINUX HOST MACHINE

The right part of Fig. 1 shows the system architecture. Currently we focus on the inference implementation. The main task of the Linux host machine is to accept the frequency domain feature vectors extracted by the FPGA board, and perform effective abnormal traffic detection to distinguish between benign and malicious traffic. Considering the realtime and speed requirements of network traffic detection, we firstly train and save a CatBoost model that focus on abnormal traffic detection in advance like Fig. 6 shows. Then we reload this pre-trained model to the CatBoost framework to achieve the realtime inference process.

1) UDP PACKET CAPTURE AND DECODE

In order to obtain the frequency domain feature vectors from the UDP packets, we utilize AF-Packet as the traffic capturing method. AF-Packet is one of the socket types which provides low-level capturing function of network packets. Relying on the PCIe driver, the UDP packets from the FPGA board enters the receiving queue of AF-Packet in realtime. Here we use ring buffer to cache the packet data. Ring buffer is a kind of data structure which is implemented with two pointers: the read pointer and the write pointer. When the writing task writes UDP packets into the ring buffer, it will put the data at the location pointed by the write pointer. Meanwhile, if the reading task reads UDP packets from the ring buffer, the data will be taken out from the location pointed by the read pointer. By moving the read and write pointers instead of moving data itself, the ring buffer implementation reduces overhead caused by data movement, thereby improving the efficiency of UDP packets capturing. At the same time, we build a UDP packet decoder to extract the frequency domain feature

vectors from the UDP payload, and send them to the next $L1$ normalization part.

2) L1 NORMALIZATION

In order to improve the performance and reliability of the machine learning model, we execute the normalization processing to the frequency domain feature vectors. Normalization is a data pre-processing approach which can change the different values of features to the standardized scale. It usually has the ability to help machine learning to obtain better performance. In our design, we adapt $L1$ normalization method to the feature vectors. The definition of the $L1$ normalization is shown as follows:

$$F(fv_x) = \frac{fv_x}{\sum_{i=1}^{n} |fv_i|} \tag{3}$$

Assuming a feature vector is defined as $[fv_1, fv_2, \ldots fv_n]$, the generated feature vector with $L1$ normalization is calculated according to the Equation 3. After that, the frequency domain feature vectors normalized by $L1$ approach will be transported to the detection module for executing further abnormal traffic detection.

3) ABNORMAL TRAFFIC DETECTION BASED ON CATBOOST

In the detection module, we use CatBoost framework [45] to perform abnormal traffic detection based on the frequency domain feature vectors. CatBoost is a machine learning library, it is a gradient boosting framework on decision trees which has improved accuracy and fast inference speed. Gradient Boosting Decision Tree (GBDT) is a machine learning algorithm used for regression and classification. It employs ensemble learning method which combines with decision trees to build an accurate and robust model. Compared with conventional GBDT algorithms, CatBoost uses optimized tree construction to reduce computational complexity and improve training speed. Meanwhile, it provides C programming interface which is convenient to integrate it with our current system. Fig. 6 indicates how we integrate the trained CatBoost model into the inference process.

In the training stage, the goal is to generate a trained model. The pre-extracted frequency domain features are sent to the CatBoost framework for training after executing $L1$ normalization. At the same time, the trained model is saved as CatBoost *cbm* format. In the inference stage, in order to minimize the overhead caused by PCIe transmission and improve the throughput, we use CPU to execute the inference process. Relying on the CatBoost shared library, the saved model can be loaded into the CatBoost inference framework in advance. In that case, the new incoming feature vectors can be predicted according to the model in realtime.

IV. EXPERIMENTAL EVALUATION

In this section, we conduct an experiment to evaluate our proposed method from training stage and inference stage.

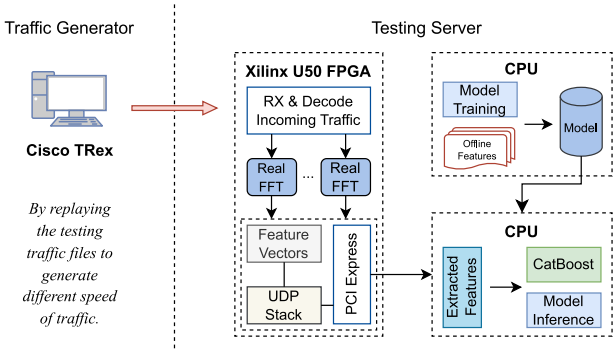


FIGURE 7. Overview of experiment environment setup.

A. EXPERIMENT ENVIRONMENT

1) SYNTHESIS AND IMPLEMENTATION

The synthesis and implementation of the frequency domain features extraction are designed by Xilinx Vivado Design Suite 2020.2.[5] We use a Xilinx Alveo U50 accelerator card as the hardware implementation, which the recourse includes 872K LUTs, 1,743K registers, 47.3Mb BRAM and 5,952 DSP slices. The software design on the Linux host machine is built with C language and it is compiled by Clang 12.

2) ENVIRONMENT SETUP

Fig. 7 shows the overview of the experiment environment setup. To evaluate the realtime performance, we adapt the Cisco TRex[6] as the traffic generator tool. Cisco TRex traffic generator runs on a separate server which is installed with a Mellanox ConnectX-5 NIC[7] to generate and send traffic. It is connected with the Xilinx U50 FPGA through the QSFP28 port. On the testing server, it is equipped with a Xilinx Alveo U50 accelerator card and an Intel Core i9-13900K CPU. The operating system is Ubuntu 20.04 LTS. The pre-trained model is firstly loaded into the CatBoost framework. When the FPGA receives the traffic sent by Cisco TRex, it will extract the frequency domain feature vectors and forward them to the CPU for realtime prediction.

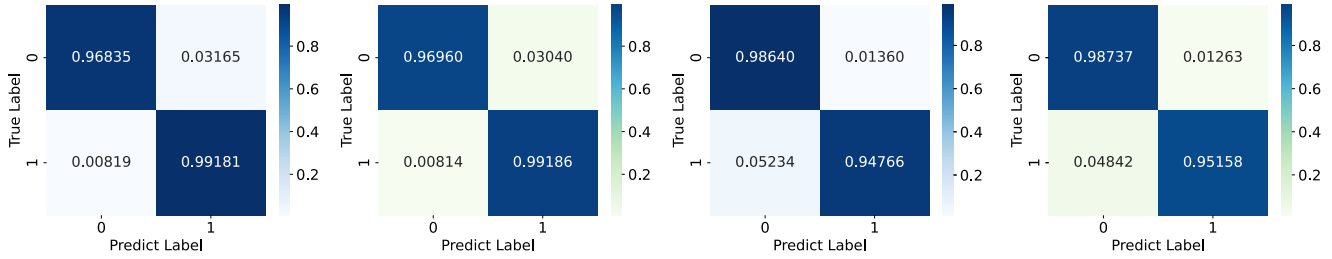
3) DATASETS

To evaluate the abnormal traffic detection performance of our proposed system, we use the MAWI WIDE dataset and the USTC-TFC2016 dataset as the experiment datasets. The MAWI WIDE dataset is created by the MAWI Working Group [43]. We choose the "samplepoint-F" traffic traces and select 1 million packets of 2022.09.11 trace as the benign traffic. The USTC-TFC2016 dataset is presented in this paper [44], it consists of different types of malware traffic which are collected from real network environment. We choose Cridex and Geodo malware traffic as the malicious traffic. We remove the traffic such as IPv6 (currently our

[5]https://docs.xilinx.com/r/2020.2-English/ug910-vivado-getting-started
[6]https://trex-tgn.cisco.com/
[7]https://www.nvidia.com/en-us/networking/ethernet/connectx-5/

TABLE 2. Experiment results before and after frequency transformation in the training stage.

Dataset	Configuration	Frequency Transformation	Accuracy	Precision	Recall	F1-score
MAWI WIDE + USTC-TFC2016 (Cridex)	Config 1	CatBoost (before)	0.97601	0.94160	0.99133	0.96583
		CatBoost (after)	**0.97656**	**0.94292**	**0.99151**	**0.96660**
	Config 2	CatBoost (before)	0.97637	0.94217	0.99181	0.96635
		CatBoost (after)	**0.97721**	**0.94432**	**0.99186**	**0.96751**
MAWI WIDE + USTC-TFC2016 (Geodo)	Config 1	CatBoost (before)	0.97855	0.92620	0.94338	0.93471
		CatBoost (after)	**0.97989**	**0.92933**	**0.94855**	**0.93884**
	Config 2	CatBoost (before)	0.98009	0.93122	0.94766	0.93937
		CatBoost (after)	**0.98155**	**0.93609**	**0.95158**	**0.94377**



(a) Before transformation (Cridex). (b) After transformation (Cridex). (c) Before transformation (Geodo). (d) After transformation (Geodo).

FIGURE 8. Normalized confusion matrixes before and after frequency transformation on Config 2.

system is designed for IPv4 traffic detection) and divide the training set and test set as the ratio of 80% and 20%.

B. TRAINING EVALUATION

In this part, we evaluate our prototype design in the training stage on the above datasets.

1) EVALUATION METRIC

The evaluation metrics that we use to evaluate the proposed system including Accuracy, Precision, Recall, F1-score and so on. The following part lists the detail descriptions.

- **False Positive (FP)** is the negative samples that are classified as positive.
- **False Negative (FN)** is the positive samples that are classified as negative.
- **Accuracy** indicates the probability of correctly predicted samples. It is defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

- **Precision** measures the probability of positive predictions. It is defined as:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

- **Recall** indicates how many positive samples have been predicted correctly. It is defined as:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

TABLE 3. Configurations of CatBoost classifier.

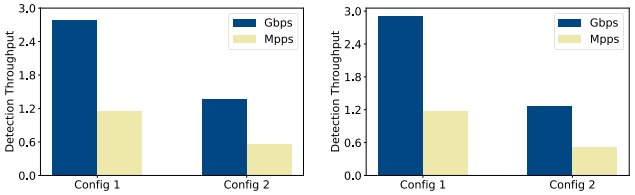
Parameter	Config 1	Config 2
depth	4	8
n_estimators	100	400

- **F1-score** is a comprehensive evaluation of precision and recall. It is defined as:

$$\text{F1-score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{7}$$

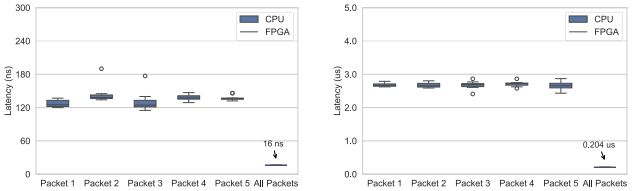
2) EVALUATION RESULTS

In order to evaluate the performance of our proposed method, we make a comparison of the CatBoost classifier before and after frequency transformation from Accuracy, Precision, Recall and F1-score on the two datasets. Table 2 shows the experiment results in the training stage. We train the CatBoost model with different configurations of depth and n_estimators according to the Table 3. From the Table 2 we can see, in the MAWI WIDE + USTC-TFC2016 (Cridex) dataset, the Accuracy, Precision, Recall and F1-score of the CatBoost classifier (Config 1) after transformation are 0.97656, 0.94292, 0.99151 and 0.96660 separately, which is better than the classifier before transformation. We also notice that with depth and n_estimators increase, our proposed method can achieve better performance that the



(a) Detection throughput (Cridex). (b) Detection throughput (Geodo).

FIGURE 9. Detection throughput with different configurations.



(a) Traffic decoding processing. (b) Frequency features extraction.

FIGURE 10. Latency of traffic decoding processing and frequency domain features extraction.

Accuracy can achieve 0.97721, the Precision can achieve 0.94432, the Recall can achieve 0.99186 and the F1-score can achieve 0.96751. Similar to the first dataset, in the MAWI WIDE + USTC-TFC2016 (Geodo) dataset, the evaluation metrics of the CatBoost classifier after transformation are 0.98155, 0.93609, 0.95158 and 0.94377 on Config 2, which shows the frequency transformation is helpful to improve the performance.

We also show the normalized confusion matrixes before and after frequency transformation on Config 2 in Fig. 8. From the Fig. 8(a) and Fig. 8(c) we can see, the normalized FP of the CatBoost classifier before transformation are separately 0.03165 and 0.01360. The normalized FN are 0.00819 and 0.05234. If we check the Fig. 8(b) and Fig. 8(d), we can find that the normalized FP of CatBoost classifier after frequency transformation are 0.03040 and 0.01263, and the normalized FN reaches 0.00814 and 0.04842 which shows the better performance on both datasets.

C. INFERENCE EVALUATION

In this part, we evaluate the proposed system from inference stage on detection throughput, latency, resource usage and power consumption.

1) DETECTION THROUGHPUT

In order to measure the detection throughput of our proposed method, we use Cisco TRex to generate different rate of testing traffic as Fig. 7 shows. We merge the benign traffic and malicious traffic to generate the testing *PCAP* file, and use Cisco TRex for traffic replay. We measure the detection throughput of our proposed method under different configurations shown in Table 3 with one single CPU core. The testing results are shown in Fig. 9.

From the results we can find, the system can reach about 1∼ 3 Gbps detection throughput even using one CPU core on both datasets. One reason is that we use FPGA to execute the time-consuming FFT which reduces the utilization of CPU resources. Another reason is that we adapt a ring buffer and C programming interface of CatBoost to reduce the cost of copying data and achieve fast inference with the loaded model. We also notice that the system of Config 1 performs better than Config 2. This is because the smaller values of depth and n_estimators reduce the inference time.

TABLE 4. Resource usage on Xilinx Alveo U50 accelerator card.

Function Block[1]	LUT	Register	BRAM	DSP
Block 1	5,103	10,155	17.5	12
Block 2	9,547	31,216	0	0
Block 3	73,258	79,981	86	0
Resource Proportion	10.1%	7.0%	7.7%	0.2%

[1] Block 1 represents FIFO & Traffic Decoder & Frequency Domain Features Extraction & UDP Stack, Block 2 represents Traffic Receiver, and Block 3 represents PCIe Module.

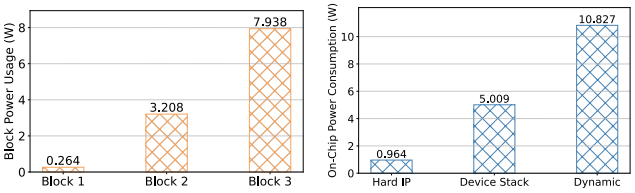
2) LATENCY

Considering the real network scenarios, we need to promptly identify malicious traffic. Therefore, ensuring realtime detection performance is crucial. Traffic decoding processing and frequency domain features extraction are time-consuming tasks that will affect realtime performance. We extract and replay five different network packets and calculate corresponding processing latency. The results of CPU and FPGA implementations are shown in Fig. 10. From the Fig. 10(a) we can see, due to some impacts such as cache or CPU scheduling, the latency of traffic decoding processing on the CPU side will fluctuate within a range. On the FPGA side, since FPGA can parse the packet header immediately when the packet arrives without waiting for the full packet including payload, the traffic decoding processing latency is a constant value which is smaller than the CPU implementation. Fig. 10(b) indicates the latency of frequency domain features extraction. We choose PyTorch[8] to execute the frequency transformation on the CPU side. From the results we can find, compared with the CPU implementation, since FPGA uses proprietary circuits to implement related operations, there is no additional overhead such as program context switching or function call, thus effectively reducing calculation latency and improving realtime detection performance.

3) RESOURCE USAGE

Due to the implementations of traffic receiving and decoding, frequency domain features extraction and transportation are inside the FPGA board, considering the resource constraints on the Xilinx Alveo U50 accelerator card, we need to evaluate the FPGA resource consumption. We calculate the resource usage on LUT, register, BRAM tile and DSP slice. The results

[8] https://pytorch.org/



(a) Power usage of different blocks. (b) On-chip power consumption.

FIGURE 11. Power consumption on Xilinx Alveo U50 accelerator card.

and respective proportion are shown in Table 4. In the Table 4, it lists the resource usage of three main function blocks. We observe that the "FIFO & Traffic Decoder & Frequency Domain Features Extraction & UDP Stack" block consumes appropriate resources. Meanwhile, from the proportion we can also observe although the PCIe module consumes a lot of LUT and register resources, the overall resource usage of our proposed system still remain relatively low across the entire resources in the FPGA board.

4) POWER CONSUMPTION

Since FPGA usually has the characteristic of high energy efficiency ratio, for this reason, we evaluate the power consumption of the FPGA design on the Xilinx Alveo U50 accelerator card. We run the implementation with power analysis by using Xilinx Power Estimator [46] to show the power usage. Fig. 11 shows the power usage on different function blocks and the on-chip power consumed by Xilinx XCU50 chip. Fig. 11(a) shows the power usage which is consumed by three main function blocks as Table 4 shows. We can observe the Block 1 occupies low power usage. Fig. 11(b) indicates the on-chip power consumption on hard IP, device stack and dynamic parts. We find that dynamic power accounts for the majority of the total on-chip power consumption. One reason is that since there is data transmission between the FPGA board and the Linux host machine, the GTY transceivers accounts for much power consumption.

V. DISCUSSION

In this section, we discuss the limitations of our current system and the advantages by using FPGA-based frequency transformation. Currently, our proposed system has the following limitations. Firstly, since we adapt a design method of offline training first and then execute realtime inference, we need to train the required model in advance and then import it into our system for prediction. This means the system cannot accept incoming traffic while conducting training operations simultaneously. Secondly, because we use CPU for training, it will slow down the training speed when the datasets are too large. At present, our research focuses on the implementation that uses frequency transformation and FPGA to enhance intrusion detection. In the future, if intrusion prevention functions (e.g., malicious traffic blocking) can be added into our system, it will be a meaningful attempt to help researchers develop more effective defense mechanism. For the advantages of our proposed system, from the experiment results in the training stage we can find, the evaluation metrics of the CatBoost classifier after frequency transformation can reach better performance. The results indicate that the frequency domain features of benign and malicious traffic that we extracted in FPGA are effective. At the same time, the feature extraction design based on FPGA implements the necessary processes from traffic receiving and decoding, frequency domain features extraction and transportation, which not only reduces the CPU usage, but also allows us to get rid of the dependence on external NICs, thus achieving efficient abnormal traffic detection.

VI. CONCLUSION

In this paper, we propose a realtime FPGA-based frequency transformation method to detect malicious traffic. An FPGA board is adapted in our system to extract and sent frequency domain features. We build a four-way parallel real transformation of FFT to calculate frequency domain feature vectors and use a UDP stack to transport them. Moreover, we use AF-Packet as the capturing method and ring buffer to cache the received UDP packets. We train a machine learning model and perform fast abnormal traffic detection by using CatBoost framework. We implement the proposed system on a Xilinx Alveo U50 accelerator card and a Linux host machine, and design an experiment to evaluate it both in the training stage and inference stage. The experimental results demonstrate that our system can reach about 0.98 detection accuracy and $1\sim 3$ Gbps realtime detection throughput. Our current system is targeted for realtime inference process. In the future, we plan to add the function of realtime online training, and utilize GPU to accelerate the training speed. We hope that through these improvements, our system can be deployed into a more universal network environment.

REFERENCES

[1] M. S. Jalali, M. Siegel, and S. Madnick, "Decision-making and biases in cybersecurity capability development: Evidence from a simulation game experiment," *J. Strategic Inf. Syst.*, vol. 28, no. 1, pp. 66–82, Mar. 2019, doi: 10.1016/j.jsis.2018.09.003.

[2] ENISA. (2020). *ENISA Threat Landscape 2020: Cyber Attacks Becoming More Sophisticated, Targeted, Widespread and Undetected*. [Online]. Available: https://www.enisa.europa.eu/news/enisa-news/enisa-threat-landscape-2020

[3] Microsoft. (2020). *Microsoft Digital Defense Report 2022*. [Online]. Available: https://www.microsoft.com/en-us/security/business/microsoft-digital-defense-report-2022

[4] F. Hussain, S. G. Abbas, G. A. Shah, I. M. Pires, U. U. Fayyaz, F. Shahzad, N. M. Garcia, and E. Zdravevski, "A framework for malicious traffic detection in IoT healthcare environment," *Sensors*, vol. 21, no. 9, p. 3025, Apr. 2021, doi: 10.3390/s21093025.

[5] K. Borders, J. Springer, and M. Burnside, "Chimera: A declarative language for streaming network traffic analysis," in *Proc. 21st USENIX Secur. Symp.*, 2021, pp. 365–379. [Online]. Available: https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/borders

[6] M. A. Jamshed, J. Lee, S. Moon, I. Yun, D. Kim, S. Lee, Y. Yi, and K. Park, "Kargus: A highly-scalable software-based intrusion detection system," in *Proc. ACM Conf. Comput. Commun. Secur.*, Oct. 2012, pp. 317–328, doi: 10.1145/2382196.2382232.

[7] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Comput. Secur.*, vol. 28, nos. 1–2, pp. 18–28, Feb. 2009, doi: 10.1016/j.cose.2008.08.003.

[8] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016, doi: 10.1109/COMST.2015.2494502.

[9] R. Tang, Z. Yang, Z. Li, W. Meng, H. Wang, Q. Li, Y. Sun, D. Pei, T. Wei, Y. Xu, and Y. Liu, "ZeroWall: Detecting zero-day Web attacks through encoder–decoder recurrent neural networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 2479–2488, doi: 10.1109/INFOCOM41043.2020.9155278.

[10] J. Wang, Q. Yang, and D. Ren, "An intrusion detection algorithm based on decision tree technology," in *Proc. Asia–Pacific Conf. Inf. Process.*, vol. 2, Jul. 2009, pp. 333–335, doi: 10.1109/APCIP.2009.218.

[11] D. M. Farid, N. Harbi, and M. Z. Rahman, "Combining Naïve Bayes and decision tree for adaptive intrusion detection," 2010, *arXiv:1005.4496*.

[12] M. Ektefa, S. Memar, F. Sidi, and L. S. Affendey, "Intrusion detection using data mining techniques," in *Proc. Int. Conf. Inf. Retr. Knowl. Manage. (CAMP)*, Mar. 2010, pp. 200–203, doi: 10.1109/INFRKM.2010.5466919.

[13] M. Panda, A. Abraham, and M. R. Patra, "Discriminative multinomial Naïve Bayes for network intrusion detection," in *Proc. 6th Int. Conf. Inf. Assurance Secur.*, Aug. 2010, pp. 5–10, doi: 10.1109/ISIAS.2010.5604193.

[14] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From throw-away traffic to bots: Detecting the rise of DGA-based malware," in *Proc. 21st USENIX Secur. Symp.*, 2012, pp. 491–506. [Online]. Available: https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/antonakakis

[15] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: Detecting botnet command and control servers through large-scale NetFlow analysis," in *Proc. 28th Annu. Comput. Secur. Appl. Conf.*, Dec. 2012, pp. 129–138, doi: 10.1145/2420950.2420969.

[16] L. Invernizzi, S. Miskovic, R. Torres, S. Saha, S.-J. Lee, M. Mellia, C. Kruegel, and G. Vigna, "Nazca: Detecting malware distribution in large-scale networks," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2014, doi: 10.14722/ndss.2014.23269.

[17] K. Bartos, M. Sofka, and V. Franc, "Optimized invariant representation of network traffic for detecting unseen malware variants," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 807–822. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/bartos

[18] M. Nasr, A. Houmansadr, and A. Mazumdar, "Compressive traffic analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Oct. 2017, pp. 2053–2069, doi: 10.1145/3133956.3134074.

[19] S. Siby, M. Juárez, C. Díaz, N. Vallina-Rodriguez, and C. Troncoso, "Encrypted DNS –> Privacy? A traffic analysis perspective," 2019, *arXiv.1906.09682*.

[20] T. van Ede, R. Bortolameotti, A. Continella, J. Ren, D. J. Dubois, M. Lindorfer, D. Choffnes, M. van Steen, and A. Peter, "FlowPrint: Semi-supervised mobile-app fingerprinting on encrypted network traffic," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2020, doi: 10.14722/ndss.2020.24412.

[21] C. Fu, Q. Li, M. Shen, and K. Xu, "Realtime robust malicious traffic detection via frequency domain analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Nov. 2021, pp. 3431–3446, doi: 10.1145/3460120.3484585.

[22] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, Dec. 2019, doi: 10.1186/s42400-019-0038-7.

[23] *Snort Rules*. Accessed: Jun. 16, 2023. [Online]. Available: https://www.snort.org/downloads#rules

[24] J. Nam, M. Jamshed, B. Choi, D. Han, and K. Park, "Haetae: Scaling the performance of network intrusion detection with many-core processors," in *Proc. 18th Int. Symp. Res. Attacks*, 2015, pp. 89–110, doi: 10.1007/978-3-319-26362-5_5.

[25] H. Li, H. Hu, G. Gu, G.-J. Ahn, and F. Zhang, "VNIDS: Towards elastic security with safe and efficient virtualization of network intrusion detection systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Oct. 2018, pp. 17–34, doi: 10.1145/3243734.3243862.

[26] N. Cascarano, P. Rolando, F. Risso, and R. Sisto, "INFAnt: NFA pattern matching on GPGPU devices," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 5, pp. 20–26, Oct. 2010, doi: 10.1145/1880153.1880157.

[27] K. Jaic, M. C. Smith, and N. Sarma, "A practical network intrusion detection system for inline FPGAs on 10 GbE network adapters," in *Proc. IEEE 25th Int. Conf. Appl.-Specific Syst., Architectures Processors*, Jun. 2014, pp. 180–181, doi: 10.1109/ASAP.2014.6868655.

[28] Z. Zhao, H. Sadok, N. Atre, J. C. Hoe, V. Sekar, and J. Sherry, "Achieving 100 Gbps intrusion prevention on a single server," in *Proc. 14th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2020, pp. 1083–1100. [Online]. Available: https://www.usenix.org/conference/osdi20/presentation/zhao-zhipeng

[29] M. A. Salama, H. F. Eid, R. A. Ramadan, A. Darwish, and A. E. Hassanien, "Hybrid intelligent intrusion detection scheme," in *Soft Computing in Industrial Applications* (Advances in Intelligent and Soft Computing), vol. 96. Berlin, Germany: Springer, 2011, pp. 295–302, doi: 10.1007/978-3-642-20505-7_26.

[30] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, "Network anomaly detection with the restricted Boltzmann machine," *Neurocomputing*, vol. 122, pp. 13–23, Dec. 2013, doi: 10.1016/j.neucom.2012.11.050.

[31] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," *EAI Endorsed Trans. Secur. Saf.*, vol. 3, no. 9, p. e2, 2016. [Online]. Available: https://dblp.org/rec/journals/sesa/JavaidNSA16.html, doi: 10.4108/eai.3-12-2015.2262516.

[32] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad, "WebWitness: Investigating, categorizing, and mitigating malware download paths," in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 1025–1040. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/nelms

[33] S. D. Anton, S. Kanoor, D. Fraunholz, and H. D. Schotten, "Evaluation of machine learning-based anomaly detection algorithms on an industrial Modbus/TCP data set," in *Proc. 13th Int. Conf. Availability, Rel. Secur.*, Aug. 2018, p. 41, doi: 10.1145/3230833.3232818.

[34] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2018, doi: 10.14722/ndss.2018.23204.

[35] A. Manna and M. Alkasassbeh, "Detecting network anomalies using machine learning and SNMP-MIB dataset with IP group," in *Proc. 2nd Int. Conf. New Trends Comput. Sci. (ICTCS)*, Oct. 2019, pp. 1–5, doi: 10.1109/ICTCS.2019.8923043.

[36] R. Kumari, Sheetanshu, M. K. Singh, R. Jha, and N. K. Singh, "Anomaly detection in network traffic using K-mean clustering," in *Proc. 3rd Int. Conf. Recent Adv. Inf. Technol. (RAIT)*, Mar. 2016, pp. 387–393, doi: 10.1109/RAIT.2016.7507933.

[37] H. Choi, M. Kim, G. Lee, and W. Kim, "Unsupervised learning approach for network intrusion detection system using autoencoders," *J. Supercomput.*, vol. 75, no. 9, pp. 5597–5621, Sep. 2019, doi: 10.1007/s11227-019-02805-w.

[38] T. Xing, D. Huang, L. Xu, C.-J. Chung, and P. Khatkar, "SnortFlow: A OpenFlow-based intrusion prevention system in cloud environment," in *Proc. 2nd GENI Res. Educ. Exp. Workshop*, Mar. 2013, pp. 89–92, doi: 10.1109/GREE.2013.25.

[39] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic DDoS defens," in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 817–832. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/fayaz

[40] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Y. Yau, and J. Wu, "Realtime DDoS defense using COTS SDN switches via adaptive correlation analysis," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 7, pp. 1838–1853, Jul. 2018, doi: 10.1109/TIFS.2018.2805600.

[41] D. Scholz, S. Gallenmüller, H. Stubbe, and G. Carle, "SYN flood defense in programmable data planes," in *Proc. 3rd P4 Workshop Eur.*, Dec. 2020, pp. 13–20, doi: 10.1145/3426744.3431323.

[42] J. Ni, W. Chen, J. Tong, H. Wang, and L. Wu, "High-speed anomaly traffic detection based on staged frequency domain features," *J. Inf. Secur. Appl.*, vol. 77, Sep. 2023, Art. no. 103575, doi: 10.1016/J.JISA.2023.103575.

[43] MAWI Working Group. *MAWI WIDE Dataset*. Accessed: Jun. 25, 2023. [Online]. Available: https://mawi.wide.ad.jp/mawi/

[44] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2017, pp. 712–717, doi: 10.1109/ICOIN.2017.7899588.

[45] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: Gradient boosting with categorical features support," 2018, *arXiv.1810.11363*.
[46] Vivado Design Suite Tutorial. (2020). *Power Analysis and Optimization*. [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2020_2/ug997-vivado-power-analysis-optimization-tutorial.pdf

ZHENGUO HU is currently pursuing the Ph.D. degree with the Graduate School of Informatics, Nagoya University. His research interests include FPGA related researches and network security.

HIROKAZU HASEGAWA (Member, IEEE) received the Ph.D. degree in information science from Nagoya University, in 2017. He was an Assistant Professor with Nagoya University, from 2017 to 2022. He has been a Project Associate Professor with the National Institute of Informatics, since 2022. His research interests include the internet and network security.

YUKIKO YAMAGUCHI received the B.S. degree in information engineering from the Nagoya Institute of Technology, in 1983, and the M.S. degree in information engineering from Nagoya University.

Since then, she was affiliated with Fujitsu Laboratories Ltd. In April 1991, she joined Nagoya University, as an Assistant Professor with the Computer Center. She is currently an Assistant Professor with the Information Technology Center. Her research interests include network management technology and cyber-security.

HAJIME SHIMADA (Member, IEEE) was born in 1976. He received the B.E., M.E., and D.E. degrees from Nagoya University, in 1998, 2000, and 2004, respectively.

He was an Assistant Professor with Kyoto University, from 2005 to 2009, and an Associate Professor with NAIST, from 2009 to 2013. He has been an Associate Professor with Nagoya University, since 2013. His current research interests include cyber-security, network operation, and computer architecture. He is a member of IPSJ and IEICE.