Heterogeneous Network Inspection in IoT Environment with FPGA based Pre-Filter and CPU based LightGBM

Zhenguo Hu
Graduate School of Informatics
Nagoya University
Nagoya, Japan
tu36rz9u@gmail.com

Hirokazu Hasegawa
Center for Strategic Cyber Resilience
Research and Development
National Institute of Informatics
Tokyo, Japan
hasegawa@nii.ac.jp

Yukiko Yamaguchi
Information Technology Center
Nagoya University
Nagoya, Japan
yamaguchi@itc.nagoya-u.ac.jp

Hajime Shimada
Information Technology Center
Nagoya University
Nagoya, Japan
shimada@itc.nagoya-u.ac.jp

***Abstract*—With the development of modern society, IoT has entered many aspects of our daily lives. At the same time, cyber attacks in IoT environments are becoming increasingly rampant. We urgently need a method to effectively inspect and detect them such as the usage of malicious traffic detection technology. Malicious traffic detection is usually divided into two aspects: signature based method and machine learning based method. The former method usually relies on pre-defined signatures or rules and cannot effectively detect unknown threats such as zero-day attacks. Although the latter method can detect unknown attacks, most of them focus on offline traffic and cannot adapt to the current realtime IoT network environment. In this paper, we propose a heterogeneous malicious traffic detection system which combines both of them to achieve the realtime detection. In this design, we utilize the bloom array to execute pre-filter in an FPGA board, and implement a CPU based LightGBM classifier to identify the filtered traffic. We also implemented an experiment to evaluate the proposed system on both training stage and inference stage, which shows the system has the ability to identify malicious traffic in the IoT network environment.**

Keywords—Malicious Traffic Detection; Machine Learning; FPGA; LightGBM

I. Introduction

With the development of information technology, Internet of Things (IoT) has gained more and more people's attractions in recent years. It improves the convenience of our lives by enabling communication between electronic devices and sensors through the Internet [1]. IoT depicts a world where anything can be connected in an intelligent fashion [2], including smart homes, smart cities, wearable devices, industrial automation and even healthcare and telemedicine. Although the development of the IoT has brought revolutionary changes to modern society, however, it also brings some security issues such as data leakage and identity theft. One of the famous IoT security threats is Mirai Botnet, it is a worm-like family of malware which changes IoT devices to DDoS botnets [3]. It was appeared as early as August 31, 2016 [4]. By scanning IoT devices on the Internet and controlling them, Mirai has the ability to launch DDoS attacks to the target network, which causes huge losses to companies and organizations. In order to protect the digital and property security of enterprises and users, and prevent the similar attacks from happening again, we need some effective defense approaches to resist and prevent the cyber threats.

Malicious traffic detection is an efficient mechanism to identify and mitigate potential threats and attacks. It is widely used in Intrusion Detection System (IDS) which is among the existing security methods responsible for detecting malicious activities [5]. Traditional malicious traffic detection usually adapts signature or feature based rules to achieve the traffic inspection process [6][7]. These methods rely on the pre-defined signatures or rules of malicious traffic such as packet protocols or payloads, which are essentially specific patterns associated with the known network attacks. However, if we just rely on the traditional signature matching mechanism, it may miss out many potential and undiscovered threats such as zero-day attacks [8][9][10]. Compared with signature based malicious traffic detection, machine learning (ML) based malicious traffic detection is a different detection approach which leverages the advantages of machine learning to detect unknown attacks and increase the detection efficiency. Therefore, in the IoT environments [11][12][13], it is useful to use machine learning methods to detect network attacks, which will apply machine learning models to learn corresponding features on the network traffic datasets. By extracting the traffic features from packet headers or payload contents, the machine learning algorithm can identify whether the network traffic is benign or malicious. Although machine learning based malicious traffic detection methods overcome some limitations of signature based detection methods, but most of them focus on the offline detection which can not deal with the realtime incoming network traffic [14][15][16].

In order to overcome this problem, in this paper, we present a realtime heterogeneous malicious traffic detection method based on LightGBM, which has the ability to detect malicious traffic with high accuracy. An FPGA based pre-filter is used to perform IP address blacklist filtering, while a packet capturer and parser is used to capture and parse the packet, and a LightGBM classifier is implemented to achieve precise detection of malicious network packets. The performance of our proposed system is evaluated from both detection speed and detection accuracy, which shows that our system can support realtime malicious traffic detection in the IoT environment.

The rest of the paper is organized as follows. In Section II, we discuss the related work about malicious traffic detection. In Section III, we introduce the prototype design of the heterogeneous malicious traffic detection system. We also conduct an experiment and evaluate the system in Section IV. Finally, we conclude this paper in Section V.

II. Related Work

In the past few years, in order to alleviate security and trust issues on the Internet, there are a lot of researches focusing their attentions on inspecting and detecting network malicious traffic. In this section, we mainly review signature based malicious traffic detection including software and hardware, and machine learning based malicious traffic detection.

A. *Signature based Malicious Traffic Detection*

On the software side, Snort [17] and Suricata [18] are the famous signature based IDS projects which are widely used in many companies or organizations. They distinguish network traffic against a set of defined signatures to identify attacks and threats. These kinds of signatures are provided by cybersecurity experts or proprietary vendors [19]. Besides, in the paper [20], J. Nam et al. propose a high-performance Suricata-based NIDS on many-core processors (MCPs) which is called Haetae to achieve traffic detection. This system adapts the parallelism of NIDS engines and uses programmable network interface cards to offload packet processing, it can also dynamically offload network traffic to host-side CPU to achieve the detection. H. Li et al. [21] design the vNIDS to offer effective detection and provisioning for NIDS virtualization by using such as detection state sharing and microservices.

On the hardware side, K. Jaic et al. [22] design a hybrid-NIDS (called SFAOENIDS), which combines the FPGA with a network interface card to provide hardware pattern matching and software post processing. In the paper [23], Z. Zhao et al. propose an FPGA-first approach called Pigasus, most of the processing and controlling of the network traffic are implemented in the FPGA to ensure the speed. Except for the FPGA components, there are also some designs which are implemented on other hardware devices. N. Cascarano et al. [24] present a regular expression matching method with a parallel engine that is implemented on GPU. T. Jepsen et al. [25] propose a string searching approach of packet payloads on a programmable network ASIC.

B. *Machine Learning based Malicious Traffic Detection*

Recently, machine learning methods (such as supervised and unsupervised learning) are widely used on malicious traffic detection. For example, in order to detect and block bot-net C&C traffic, M. Antonakakis et al. [14] propose a method called Pleiades to identify randomly generated domains without reversing. They combine clustering and classification algorithms, monitor traffic below the local recursive DNS server and analyze streams of unsuccessful DNS resolutions to achieve detection process. In the paper [26], T. Nelms et al. present the WebWitness, which is an incident investigation system to trace back events before malware downloads happening, and leverages the paths to build effective defenses. They also deploy their system on an academic network to collect malicious download paths, which shows the system can successfully decrease the infection rate. In addition, in order to resist the threat of malware, L. Invernizzi et al. [16] propose a system called Nazca which identifies infections in large scale networks by investigating how a client downloads and installs malware in the real world. Through checking the telltale signs of the malicious network infrastructures, Nazca has the ability to detect previously-unseen malware and can not be easily influenced by code obfuscation. Y. Mirsky et al. [27] also present another approach called Kitsune which uses neural networks to distinguish normal traffic and network attacks. This method adapts the ensemble of neural networks which are called autoencoders to identify the benign and abnormal network traffic. They also propose a dataset named Mirai which includes the real network information of the Mirai botnet malware.

III. Heterogeneous Malicious Traffic Detection System Design

The heterogeneous malicious traffic detection system is designed to detect packet-level malicious traffic in IoT environment, it mainly consists of two parts: FPGA based Pre-Filter and Machine Learning based Traffic Detection. The former is used to preliminary filter the incoming traffic, while the latter is used for further traffic detection. Figure 1 shows the overview of the system architecture.

A. *FPGA based Pre-Filter*

1) Description of FPGA based Pre-Filter: In a network environment, there will be some truly malicious traffic with specific characteristics such as IP address. These traffics can be blocked by setting the software firewall and corresponding rules. Compared with the traditional firewall that uses CPU to handle traffic, an FPGA based pre-filter can be programmed with specific rules to filter or discard data with low resource consumption and high performance. At the same time, it can implement underlying protocol parsing (such as Layer 2, Layer 3) to achieve more accurate detection. Currently, we execute

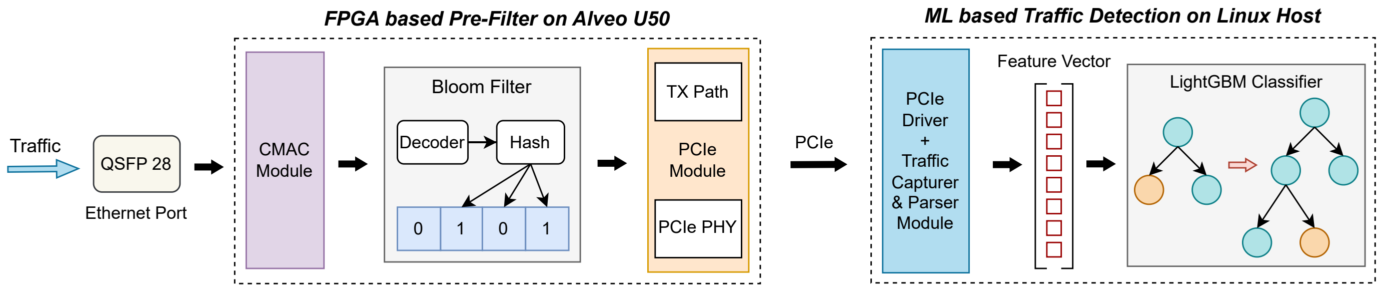


Figure 1. Overview of the system architecture.

“Source IPv4 Address” matching on the third layer of the OSI model. The main function of this pre-filter is to filter the truly malicious traffic by setting a blacklist in the FPGA board. When the irregular traffic from some IP addresses come in, it will drop these packets automatically.

In order to achieve end-to-end packet processing and detection, the FPGA based pre-filter consists of four components which are shown as follows. We implement the prototype design based on AMD OpenNIC project [28].

- Ethernet Port: Ethernet port is a physical component (such as QSFP 28 interface) which can create an ethernet connection with the network to be detected. It is used to forward the traffic to the CMAC module.
- CMAC Module: CMAC module is used to receive the traffic packet. Here we adapt the Xilinx CMAC IP core as the implementation.
- Bloom Filter: After getting the packet, the bloom filter will decode it and extract the “Source IPv4 Address” as the matching data. The result will indicate whether this packet can be reserved or blocked.
- PCIe Module: The packets that flow out through the bloom filter will be sent to the ML based traffic detection for further inspection through the PCIe interface. It relies on the PCIe driver to achieve the communication with the host machine.

2) Bloom Filter: Bloom filter is a space-efficient data storage mechanism to decide whether an element is located in a data set. Figure 2 shows the architecture. It utilizes a bit array and hash functions to store information of the element states. In our design, we use it as the IP blacklist implementation and the elements in bloom array map the “Source IPv4 Address”. In order to add an element to the filter, it needs the hash computations. We adapt the CRC hash as the hash functions, and each hash function calculates the element location in the bloom array. When one element comes from the CMAC module, to check if it is in the bloom array, the same hash functions are applied. If any of the calculated positions are not 1, it means this element is not in the defined array.

$$\text{False Positive Probability} = \left(1 - e^{-\frac{kn}{m}}\right)^k \qquad (1)$$

Another property of the bloom filter is that there is a false positive probability. Hash collision is one reason that causes

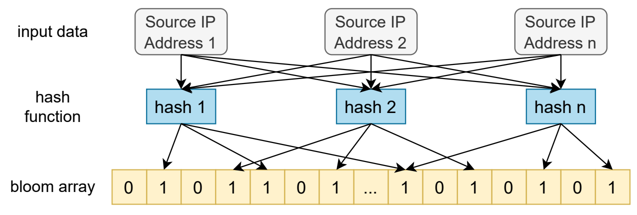


Figure 2. Architecture of the bloom filter.

the false positive probability. According to the equation (1), k is the number of hash functions, n is the number of elements, and m is the size of the bloom array. We choose $k=4$, $m/n=16$, where the false positive probability is computed to 0.239% that meets our design goal.

B. Machine Learning based Traffic Detection

Machine learning based traffic detection leverages the data-driven insight ability of machine learning to analyze the malicious traffic on the CPU side. Compared with the traditional signature based IDS systems, it has the adaptability and flexibility and can even detect certain undiscovered attacks. In order to ensure the realtime performance and detect the attack behaviors timely, we implement two modules including the Traffic Capturer and Parser Module, and the LightGBM Classifier that are shown in Figure 1. We focus on the implementation of the inference process.

1) Traffic Capturer and Parser Module: After filtering the traffic using FPGA based pre-filter, we need to capture and parse it for further detection. We adapt the *libpcap* implementation to execute the realtime capturing. It builds a connection between the FPGA board and the software stack. By capturing from the FPGA board, we can get the corresponding packet-level filtered traffic. After that, we need to analyze and pre-process each packet to get the packet features (e.g., IPv4 Length). We mainly adapt the header information instead of the content to improve the universality of our system. At the same time, considering we need to replay the traffic to achieve realtime detection, we do not use the time related features such as timestamp. We perform hierarchical splitting on each data packet and extract corresponding features according to the needs of detection. The specific features that we use and their descriptions are shown in the Table I.

TABLE I
FEATURE DESCRIPTION

Feature	Description
IPv4 Length	The length of an IPv4 packet
IPv4 ID	The identification of an IPv4 packet
IPv4 TTL	The time to live of an IPv4 packet
Layer	The type of protocol
Source Port	The source port
Destination Port	The destination port
Source IP Address	The source IP address
Destination IP Address	The destination IP address

We focus on the packet-level feature extraction. Through detecting each data packet, we can timely identify the malicious traffic. Among these features, since some of them have different categories which can not be sent into the machine learning model directly. In that case, we encode the label (such as the LabelEncoder encoding in scikit-learn) for Layer, Source IP Address and Destination IP Address to convert the category format into the number format. The processed features will be combined into a feature vector and will be transported into the LightGBM classifier for inspection.

2) LightGBM Classifier: LightGBM is a gradient boosting decision tree (GBDT) framework which is widely used for both classification and regression problems. It has advantages of high accuracy, speed and scalability. Compared with other traditional GBDT algorithms, it adapts some optimization technologies (e.g., Histogram Decision Tree, Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB)) to accelerate the training speed. The processing flow of a realtime classifier based on LightGBM is shown as follows: Firstly, import the pre-trained LightGBM model and instantiate it. The model is saved in *txt* file format in advance. Secondly, the extracted feature vectors are converted into the floating point array and fed into the instantiated LightGBM model. Next, the model conducts inference calculations to obtain the classification probability of the feature vector. Finally, we determine whether the data packet is benign or malicious based on its inference result.

IV. EXPERIMENT AND EVALUATION

We designed an experiment to evaluate the prototype malicious traffic detection system with a specific IoT network attack dataset. We also evaluated our system from two stages: training stage and inference stage.

A. Dataset

The experiment and evaluation dataset that we use comes from Kitsune Mirai [27]. It is presented in 2018 and it is captured from an IoT network, where the Mirai malware begins to infect other devices and scans for new victims network. Table II shows the details of the experiment dataset. It consists of 642,516 pieces of malicious data and 121,621 pieces of benign data. We selected 80% (611,309) of them as the training set and 20% (152,828) as the test set.

TABLE II
DETAILS OF THE EXPERIMENT DATASET

Name	Year	Number	
Kitsune Mirai	2018	***Num of Malicious***	***Num of Benign***
		642,516	121,621
		Num of Train	***Num of Test***
		611,309	152,828

B. Evaluation on Training Stage

The evaluation on the training stage can reflect the accuracy of our method. The specific details of the evaluation metrics are shown as follows.

1) Evaluation Metric: We use the following evaluation metrics to evaluate the performance of our system:

- **False Positive (FP)** represents the negative samples predicted to be positive.
- **False Negative (FN)** represents the positive samples predicted to be negative.
- **Accuracy (ACC)** is adapted to evaluate the overall performance of our system:

$$\text{ACC} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

- **Precision** is used to measure the accuracy of positive predictions:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

- **Recall** evaluates the ability of the model to detect positive samples:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

- **F1-score** is a harmonic mean between the precision and the recall:

$$\text{F1-score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}$$

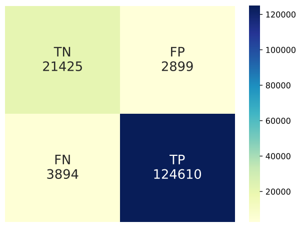
2) Evaluation Results: Table III shows the comparison of evaluation results on different classifiers. We set four other models including Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Random Forest and Decision Tree as the comparisons. From the results, we can get that using the packet-level features extracted from traffic achieves good detection performance, which proves that the features in Table I are effective. At the same time, we observe that LightGBM achieves the highest evaluation in all the classifiers on ACC, Precision, Recall and F1-score. It reaches 0.9589, 0.9774, 0.9736 and 0.9755 respectively. This is one of the reasons that we choose LightGBM as the classifier in the ML based traffic detection. We also make a comparison between Decision Tree and LightGBM models with heatmap representations in Figure 3 to indicate the metrics of FP and FN, which shows that our method has a better performance than other approaches.

C. Evaluation on Inference Stage

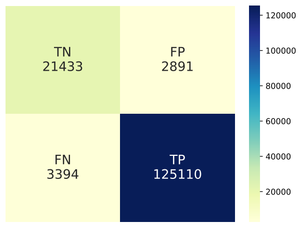
The evaluation on the inference stage can reflect the realtime detection capability of our system. It is separated into two aspects: FPGA Resource Utilization and Throughput.

TABLE III
COMPARISON OF EVALUATION RESULTS

Classifier	ACC	Precision	Recall	F1-score
SVM	0.9104	0.9706	0.9214	0.9453
KNN	0.9494	0.9740	0.9656	0.9698
Random Forest	0.9525	0.9766	0.9668	0.9716
Decision Tree	0.9556	0.9773	0.9697	0.9735
LightGBM	**0.9589**	**0.9774**	**0.9736**	**0.9755**



(a) Decision Tree



(b) LightGBM

Figure 3. Heatmaps between different methods.

1) Experiment Environment: Table IV shows the assumed experimental IoT network according to the Kitsune Mirai dataset. The number of clients is calculated by the destination mac address. By adjusting the bloom array in the FPGA based pre-filter, we build four situations of irregular traffic which includes different types of source IPv4 addresses. The experiment environment is shown in Figure 4, we use Cisco TRex [29] as the traffic generator. On the testing server, the Xilinx Alveo U50 accelerator card [30] is our FPGA platform which has 872K LUTs, 1743K registers and 47.3 Mb BRAM. The Linux host machine is installed with an Intel i9-13900K cpu and 64GB memory. The filtered traffic will be forwarded to the ML based traffic classifier.

TABLE IV
ASSUMED EXPERIMENTAL IOT NETWORK

Source IPv4 Address Range	Clients	Assumed Irregular Traffic filtered by FPGA based Pre-Filter	
		Source IPv4 Address	*Situations*
192.168.2.0/24 and 0.0.0.0	30	192.168.2.108	Situation1
		192.168.2.108 192.168.2.1	Situation2
		192.168.2.108 192.168.2.1 192.168.2.113	Situation3
		192.168.2.108 192.168.2.1 192.168.2.113 192.168.2.110	Situation4

2) FPGA Resource Utilization: The resource utilization refers to the usage of hardware resources on an FPGA board. Table V indicates the consumption of LUT, Register and BRAM Tile which comes from Xilinx Vivado Design Suite 2020.2, and shows the respective proportion in Xilinx U50 accelerator card. From the result we can see, the filter module consumes 5,279 LUTs and 5,209 registers which occupies a reasonable resource consumption.

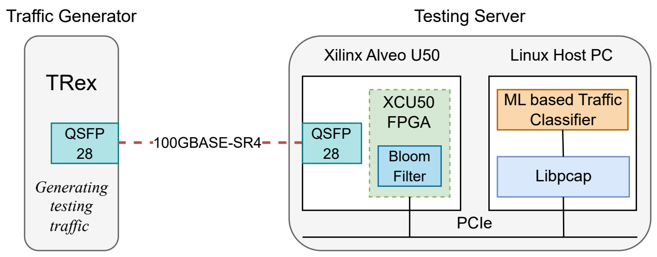


Figure 4. Overview of the experiment environment.

TABLE V
RESOURCES CONSUMPTION IN XILINX U50

Module Name	LUT	Register	BRAM Tile
CMAC Module	9,793	31,504	0
Filter Module	5,279	5,209	0
PCIe Module	79,576	84,544	94
Proportion	10.9%	7.0%	7.0%

3) Throughput: We adjust the time interval of packets in TRex and replay the Kitsune Mirai dataset to measure the throughput. In our FPGA design, the data bus width is set to 512-bit and the clock frequency is set to 250MHz. Since we adapt the pipeline design, the throughput of the pre-filter is calculated as following:

$$\text{Throughput} = \frac{512\text{-}bit}{(1/250MHz) * 10^9} = 128Gbps \qquad (6)$$

Considering the design throughput of the Xilinx CMAC IP core is 100Gbps, the calculated throughput of the pre-filter has the ability to reach the line speed of 100Gbps.

Figure 5 shows the experiment results of the complete system. The horizontal axis represents the different situations in Table IV where truly malicious traffic will be filtered by FPGA based pre-filter, while the vertical axis represents the detection throughput. *Baseline* indicates that there are no rules being enabled in the FPGA based pre-filter. From this figure we can see, the FPGA based pre-filter can block a portion of malicious traffic by setting up an IP blacklist which reduces the burden on ML based traffic detection, and effectively improve the overall detection throughput of the system. At the same time, we also observe the current system performance

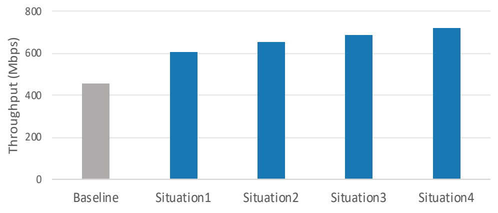


Figure 5. Throughput under different situations.

is limited with LightGBM side. If we improve LightBGM classifier part such as increasing calculation cores or changing to other packet capture methods, it still has the ability to improve the performance of the entire system.

V. Conclusion

In this paper, we present a realtime heterogeneous malicious traffic detection method based on LightGBM classifier especially for IoT environment. We built an FPGA based pre-filter to filter the truly malicious traffic through the bloom array. A traffic capturer and parser are used to receive and extract features from the filtered traffic, while a CPU based LightGBM classifier is responsible for inspecting and detecting the traffic in realtime. In order to evaluate the prototype design, we used a malicious traffic dataset of the IoT malware to test the proposed system from both training stage and inference stage. The results on the training stage show that our method has better performance than the traditional machine learning methods. Moreover, we built an experiment environment with the traffic generator and testing server to evaluate this system on inference stage. The results indicate that our system has a low FPGA resource usage and effective throughput improvement. In future work, we will explore to add other models to our system to further improve the detection efficiency.

Acknowledgment

This research was partially supported by "Nagoya University Interdisciplinary Frontier Fellowship" supported by Nagoya University and JST, the establishment of university fellowships towards the creation of science technology innovation, Grant Number JPMJFS2120, and the MEXT/JSPS KAKENHI, Grant Number JP23H03396.

References

[1] S. Kumar, P. Tiwari and M. Zymbler, "Internet of Things is a revolutionary approach for future technology enhancement: a review," J. Big Data, Vol. 6, pp. 111, 2019.

[2] S. Madakam, R. Ramaswamy and S. Tripathi, "Internet of Things (IoT): A Literature Review," Journal of Computer and Communications, Vol. 03, pp. 164-173, 2015.

[3] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C Seaman, N. Sullivan, K. Thomas and Yi Zhou, "Understanding the Mirai Botnet," In 26th USENIX Security Symposium (USENIX Security 2017), pp. 1093–1110, 2017.

[4] @unixfreaxjp, "Mmd-0056-2016 - Linux/Mirai, how an old ELF malcode is recycled", http://blog.malwaremustdie.org/2016/08/mmd-0056-2016-linuxmirai-just.html.

[5] F. Hussain, S. G. Abbas, G. A. Shah, I. Miguel 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, pp. 3025, 2021.

[6] K. Borders, J. Springer and M. Burnside, "Chimera: A Declarative Language for Streaming Network Traffic Analysis," In Proceedings of the 21th USENIX Security Symposium, pp. 365-379, 2012.

[7] 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 Proceedings of the ACM conference on Computer and Communications Security (CCS '12), pp. 317-328, 2012.

[8] 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, No. 1-2, pp. 18-28, 2009.

[9] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," IEEE Commun. Surv. Tutorials, Vol. 18, No. 2, pp. 1153-1176, 2016.

[10] 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 Proceedings of the 39th IEEE Conference on Computer Communications (INFOCOM 2020), pp. 2479-2488, 2020.

[11] D. Ventura, D. C. Mansilla, J. Lopez-de-Armentia, P. Garaizar, D. López-de-Ipiña and V. Catania, "ARIIMA: A Real IoT Implementation of a Machine-Learning Architecture for Reducing Energy Consumption," In Proceedings of the 8th International Conference on Ubiquitous Computing and Ambient Intelligence, pp. 444-451, 2014.

[12] R. Xue, L. Wang and J. Chen, "Using the IoT to Construct Ubiquitous Learning Environment," In Proceedings of the Second International Conference on Mechanic Automation and Control Engineering, pp. 7878-7880, 2011.

[13] M. A. Alsheikh, S. Lin, D. Niyato and H. P. Tan, "Machine Learning in Wireless Sensor Networks: Algorithms, Strategies, and Applications," IEEE Commun. Surv. Tutorials, Vol. 16, No. 4, pp. 1996-2018, 2014.

[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 Proceedings of the 21th USENIX Security Symposium, pp. 491-506, 2012.

[15] L. Bilge, D. Balzarotti, W. K. Robertson, E. Kirda and C. Kruegel, "Disclosure: Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis," In Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC 2012), pp. 129-138, 2012.

[16] L. Invernizzi, S. Miskovic, R. Torres, C. Kruegel, S. Saha, G. Vigna, S. Lee and M. Mellia, "Nazca: Detecting Malware Distribution in Large-Scale Networks," In Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS 2014), 2014.

[17] Snort, https://www.snort.org/.

[18] Suricata, https://suricata.io/.

[19] Snort Rules, https://www.snort.org/downloads#rules.

[20] J. Nam, M. Jamshed, B. Choi, D. Han and K. Park, "Haetae: Scaling the Performance of Network Intrusion Detection with Many-Core Processors," In Proceedings of the 18th International Symposium on Research in Attacks, Intrusions, and Defenses, pp. 89-110, 2015.

[21] H. Li, H. Hu, G. Gu, G. Ahn and F. Zhang, "VNIDS: Towards Elastic Security with Safe and Efficient Virtualization of Network Intrusion Detection Systems," In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), pp. 17-34, 2018.

[22] K. Jaic, M. C. Smith and N. Sarma, "A Practical Network Intrusion Detection System for Inline FPGAs on 10GbE Network Adapters," In Proceedings of the 25th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP 2014), pp. 180-181, 2014.

[23] Z. Zhao, H. Sadok, N. Atre, J. C. Hoe, V. Sekar and J. Sherry, "Achieving 100Gbps Intrusion Prevention on a Single Server," In Proceedings of the 14th USENIX Conference on Operating Systems Design and Implementation (OSDI'20), pp. 1083-1100, 2020.

[24] N. Cascarano, P. Rolando, F. Risso and R. Sisto, "INFAnt: NFA Pattern Matching on GPGPU Devices," Comput. Commun. Rev., Vol. 40, No. 5, pp. 20-26, 2010.

[25] T. Jepsen, D. Alvarez, N. Foster, C. Kim, J. Lee, M. Moshref and R. Soulé, "Fast String Searching on PISA," In Proceedings of the 2019 ACM Symposium on SDN Research (SOSR '19), pp. 21-28, 2019.

[26] T. Nelms, R. Perdisci, M. Antonakakis and M. Ahamad, "WebWitness: Investigating, Categorizing, and Mitigating Malware Download Paths," In Proceedings of the 24th USENIX Conference on Security Symposium, pp. 1025-1040, 2015.

[27] Y. Mirsky, T. Doitshman, Y. Elovici and A. Shabtai, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," In Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS 2018), 2018.

[28] AMD OpenNIC, https://github.com/Xilinx/open-nic.

[29] Cisco TRex, https://trex-tgn.cisco.com/.

[30] Xilinx Alveo U50 Data Center Accelerator Card, https://www.xilinx.com/products/boards-and-kits/alveo/u50.html.