In-network DDoS detection and mitigation using INT data for IoT ecosystem

Gereltsetseg Altangerel* and Máté Tejfel**

***Abstract*—Due to the limited capabilities and diversity of Internet of Things (IoT) devices, it is challenging to implement robust and unified security standards for these devices. Additionally, the fact that vulnerable IoT devices are beyond the network's control makes them susceptible to being compromised and used as bots or part of botnets, leading to a surge in attacks involving these devices in recent times. We proposed a real-time IoT anomaly detection and mitigation solution at the programmable data plane in a Software-Defined Networking (SDN) environment using In-band Network telemetry (INT) data to address this issue. As far as we know, it is the first experiment in which INT data is used to detect IoT attacks in the programmable data plane. Based on our performance evaluation, the detection delay of our proposed approach is much lower than the results of previous Distributed Denial-of-Service (DDoS) research, and the detection accuracy is similarly high.**

***Index Terms*—IoT anomaly detection, data plane, In-band Network Telemetry(INT)**

I. Introduction

With the growth of IoT usage and the ease of exploiting vulnerable IoT devices, the flow of IoT-based attacks has reached unprecedented levels [1], [2]. For example, Mirai is one of the most well-known IoT attacks. It targets insecure IoT devices and turns them into a massive botnet that can be used to launch powerful DDoS attacks. In 2016, Mirai attack on Dyn DNS (Domain name service) provider took down high-profile websites and services such as GitHub, Reddit, Netflix, and Airbnb.

One of the main reasons for the increase in IoT attacks is that organizations may not always have complete control over IoT devices that are located outside of their scope or not directly accessible. This could include situations where the organization operates in a shared office building or public space, where IoT devices may be installed by other tenants or individuals and are not managed by the organization. Despite this challenge, organizations can take measures to reduce the risks associated with these external IoT devices. These measures can be Network Segmentation and Firewalls, Monitoring and Anomaly Detection, Secure IoT Protocols, and Azure IoT Hub (or similar solutions). For example, the last solution enables secure communication between IoT devices and cloud applications while providing the ability to revoke access to unauthorized devices.

In this paper, we proposed an IoT monitoring, anomaly detection and mitigation solution for this issue. Our proposed solution is an in-network (data plane) approach for detecting and mitigating DDoS-like attacks in an IoT environment using INT data. DDoS is a common type of computer network attack that can be easily carried out using insecure IoT devices. INT is a new type of monitoring mechanism that can collect more detailed network information (INT data) in real-time than conventional monitoring, thereby helping to detect not only IoT but also other types of attacks. As far as we know, all previous works on IoT anomaly detection have used datasets based on network traffic. As IoT devices pose challenges to the implementation of standard security solutions, we aimed to create a solution in the SDN environment.

Our proposed method has two main advantages. First, the detection is faster since it is performed directly on the data processing path, and second, it is more efficient since it uses real-time INT data. The main contributions of this work are as follows.

- Generating a new INT dataset under normal and DDoS attack conditions in an SDN simulation environment and making it publicly available to fill research gaps.
- We are the first to propose a P4-based (data plane-based) method for DDoS detection and mitigation in an IoT environment using INT data.
- Evaluating the performance of the proposed method and conducting a comparison with a competing solution [3].

The rest of the paper is organized as follows. Section II is about related works, and Section III describes how we created an experimental network and collected INT data. Section IV and V describe our proposed model, the experimental results, and discussions. Finally, we summarize our work and suggest future directions in Section VI.

II. Related work

Several significant research works [4], [5], [6], [7], [8] have been proposed for IoT anomaly detection, with and without SDN. For instance, Del-IoT [9] is an IoT anomaly detection approach that employs a deep ensemble model to address data imbalance issues in network traffic datasets. It is implemented on an SDN controller. Bhunia and Gurusamy [7] present a machine learning-based anomaly detection and mitigation method for IoT traffic using SDN. They utilize the SVM algorithm to monitor and learn the behavior of IoT devices over time to detect anomalies.

In the current landscape of IoT anomaly detection studies [10], [11] controller-based anomaly detection methods are prevalent in SDN networks. The majority of these solutions leverage network traffic datasets (e.g., NetFlow, Wireshark)

The authors are with the Department of Programming Languages and Compilers, Eötvös Loránd University (ELTE), Budapest, Hungary. (e-mail: * gereltsetseg@inf.elte.hu; ** matej@inf.elte.hu).

DOI: 10.36244/ICJ.2023.5.8

with machine learning and deep learning techniques for anomaly detection.

Recent efforts in network applications have focused on reducing processing delays by offloading tasks from the control plane to the data plane or dedicated processors [12], [13]. Implementing anomaly detection directly in the packet processing path or data plane can significantly enhance detection speed compared to control plane solutions.

Since programmable data plane is a relatively new concept, data plane-based anomaly detection solutions are less common compared to controller-based solutions. However, there are some anomaly detection solutions specifically designed for the data plane. For example, Euclid [3] is a data plane-based DDoS detection solution that employs Shannon entropy based on the frequency of source and destination IP addresses. Sanghi et al. [14] focus on identifying potential attacks on data plane systems and present a scalable tool for real-time detection. Kang et al. [15] propose an approach to discover attack vectors in a data plane system and conduct preliminary experiments to demonstrate its feasibility. Although these solutions represent pioneering attempts to detect anomalies in the data plane, none of them utilize INT data.

To the best of our knowledge, there are two experimental solutions that leverage INT data for anomaly detection. Kim et. al. [16] uses a recurrent neural network (RNN), while our earlier work [10] utilizes a one-dimensional Convolutional neural network (1D CNN). However, both of these solutions are implemented on external controllers or servers. The main distinctive feature of our proposed solution in this paper is its ability to detect IoT anomalies directly on the data plane using INT data, offering a unique approach in this domain.

III. INT DATA COLLECTION

A. *Overview of the programmable data plane and INT*

Packet processing algorithms in network architectures are categorized into control and data planes based on their functions. Data plane algorithms define the packet processing pipeline, while control plane algorithms set the packet processing rules. The interaction between the control plane and the data plane is shown in Fig. 1a.

In traditional network architecture, both types of algorithms are preconfigured on each network device and are not easily customizable. Only device manufacturers have the capability to reprogram them. However, with the advent of the SDN paradigm, the control plane is decoupled from the data plane and operates on dedicated server(s). This separation has led to increased flexibility, manageability, openness, and programmability in computer networks [17].

The concept of the programmable data plane is relatively new compared to the programmable control plane. It enables anyone to quickly design, test, and deploy a variety of applications in the data plane.

P4 is one of the popular domain-specific programming languages used for defining data plane algorithms. The basic architecture of the P4 pipeline, as shown in Fig. 1b, consists of three main parts: the parser, match-action, and deparser.

1) **Parser:** The parser receives incoming packets and extracts header fields from them. This step involves parsing the packet's structure to identify and extract relevant information.
2) **Match-Action:** The match-action section processes the packet headers and metadata. It comprises one or more tables, with each table having a key part and an action part. During table application, the program attempts to find the most suitable key in the table based on the packet's header fields and metadata. Upon finding the appropriate key, the associated action is executed. If there is no matching key, the program either executes the default action if defined or does nothing. An example of a commonly used routing table in P4 involves the match key being the destination IP address. The match type can be the longest prefix match, and the corresponding action depends on the match result, such as forwarding the packet, dropping it, or applying no action.
3) **Deparser:** The deparser assembles the processed header fields and the original payload to build the outgoing packet. This step ensures the proper formatting and structuring of the packet before it leaves the network device [18].

Overall, data plane programmability with P4 or any other language empowers network administrators and developers to define customized data plane behaviors, offering greater flexibility and control over how network packets are processed and forwarded. Moreover, it allows for the development of many interesting applications [12].

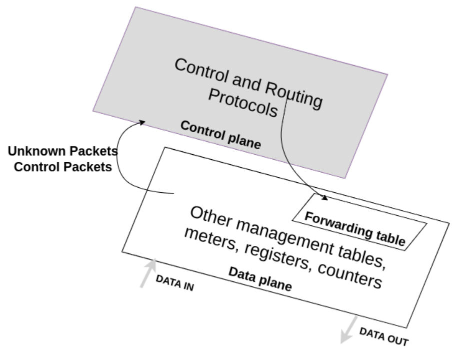
One notable application enabled by data plane programmability is INT. It is a new monitoring system that captures network telemetry information, such as hop latency, flow latency, and queue depth, directly from the data plane. The distinct advantage of this approach is that it bypasses the CPU-driven control plane, resulting in more real-time collection of telemetry data (INT data) compared to traditional monitoring mechanisms [16]. The real-time monitoring capability offered by INT provides valuable insights into network performance. This data enables network administrators to optimize network efficiency and performance while developing effective methods such as network anomaly detection, smart congestion control or routing mechanisms based on these insights.

B. *INT data collection on testbed network*

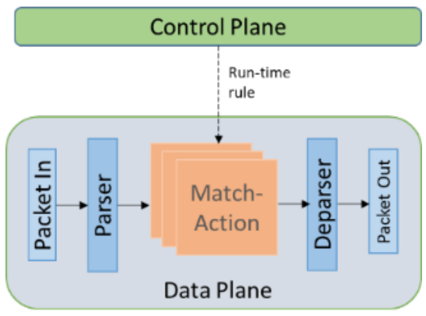
We created an INT-enabled SDN network, as shown in Figure 2, on the MININET[1] simulation program. This SDN network consists of a data plane, a control plane, IoT servers, and users. In the data plane, we deployed BMv2 software switches and configured P4 pipeline with INT support. For the control plane, we developed a custom controller using Python. This controller is responsible for configuring the packet processing rules and control rules for the data plane, providing the necessary instructions to the switches.

Within this simulated network, we emulated external IoT devices as attacker nodes and servers as target nodes. This

[1] http://mininet.org/



(a) Control & Data Plane Interaction.



(b) Abstract Packet Forwarding in P4.

Fig. 1. Network packet processing system.

setup allows us to analyze the behavior of the IoT anomalies and test the effectiveness of our proposed INT-based anomaly detection and mitigation approach.

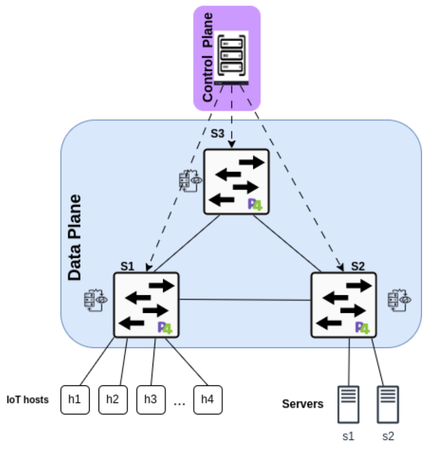


Fig. 2. Testbed SDN network.

After building a testbed network, we collected INT data including the queue depth of the switches' output interface under normal and attack conditions. To collect the INT data, we designed our own probe packet, which is a specialized packet transmitted across the network every millisecond from a specific source to a specific destination. As the probe packet is processed by the switches in the network, each switch appends its queue depth value to the packet. This action allows us to capture real-time queue depth value at each switch in the network. When the probe packet reaches its destination, we extracted queue depth information collected from each switch and stored it in a text file. As mentioned above, we created two kinds of data collection scenarios.

In the first scenario, we generated legitimate traffic, including UDP and ICMP flows using iperf[2] and ping[3] tools from external IoT devices to the target server. During these normal conditions, we captured INT data, including queue depth information, using the probe packet. Over the course of an hour, we accumulated more than 5000 data points, which were saved to a text file for analysis.

In the second scenario, we collected INT data under malicious traffic conditions. Due to hardware limitations, we need lighter attack. Therefore, we chose DDoS attacks with ICMP flooding. We created a DDoS attack using the hping3[4] tool, and legitimate traffic flows with UDP and ICMP messages from the attacker node to the target node. INT data were also collected for one hour under this condition.

C. Characteristics of INT data

After collecting INT data, we conducted a statistical analysis to understand their behavior. Firstly, we performed a t-test to evaluate and compare the difference between the means of the normal and malicious INT datasets, each consisting of 5000 samples. The result is presented in the first column of Table 1. A negative T-statistic (-33.74) suggests that the mean of the normal group is lower than that of the malicious group in our case. The P-value quantifies the probability of obtaining the observed results under the assumption of the null hypothesis. An extremely small P-value (e.g., 3.55e-181) suggests strong statistical significance and it indicates that there is substantial evidence to reject the null hypothesis in favor of the alternative hypothesis, supporting the presence of a meaningful and significant difference between the means of the two compared groups. In summary, based on this t-test result, we can conclude that the normal and malicious datasets were statistically significantly different at a very high probability.

Additionally, we created smaller datasets by randomly selecting 128 records multiple times from the original 5000-sample datasets of both malicious and normal conditions. The t-test results for these smaller datasets are shown in columns 2 to 4 of Table 1. These results were consistent with the previous whole dataset analysis, confirming the significant differences between the two conditions.

Furthermore, we compared the datasets using simple statistical measures like mean, median, mode, etc. Based on these measures, significant differences were observed among the datasets, indicating the potential to detect IoT anomalies

[2] https://iperf.fr/
[3] https://linuxize.com/post/linux-ping-command/
[4] https://www.kali.org/tools/hping3/

TABLE I
T-TEST RESULTS: STATISTICAL COMPARISON OF TWO DATASETS

	T-statistic	P-value
Whole dataset	-33.74	3.55e-181
Sample 1	-9.48	8.63E-18
Sample 2	-7.31	6.61E-12
...	...	...
Sample n	-9.628	3.15E-18

effectively. Since our anomaly detection solution for the IoT ecosystem aims to be simple and fast based on P4-language capabilities, we decided to implement it based on the mean value of queue depth.

IV. PROPOSED P4-BASED ANOMALY DETECTION AND MITIGATION APPROACH

After collecting and analyzing the INT data, we developed a P4-based packet processing pipeline incorporating IoT anomaly detection and mitigation mechanisms. Our proposed P4-based approach, depicted in Fig. 3, operates based on three distinct states: normal, detection, and mitigation state. Each state is designed with specific functionalities, and transitions between these states occur based on predefined conditions. We used global register and metadata in P4 to create a state that can be accessed and manipulated from both the data plane and the control plane. As a result, state transitions are effectively orchestrated by both the data plane and control plane according to the configuration specified in Table II.

In the subsequent text, the block numbers in parentheses provide a reference to how the numbered blocks of the P4 pipeline depicted in Fig. 3 correspond to the descriptions provided.

First of all, the packet counter value is analyzed to determine whether to maintain the current normal state or switch to the anomaly detection state. Packet counters are implemented at the ingress part of the pipeline in the data plane and are only read by the control plane. Based on the configured threshold counter value, the control plane will set either of the two states mentioned above. In order to determine a baseline for the packet counter, we conducted a test to determine the number of packets and bytes transmitted through the input and output interfaces of the S1 switch (which attackers are connected to) over a period of 2 seconds, while transmitting normal and attack packets. We then computed the mean value from 3000 samples of packet counter for each interface of S1, which were subsequently used as baselines to determine whether to initiate the detection state.

In the default (normal) state, the packets are handled according to the white blocks in Fig. 3. The main function in this state is forwarding the packet based on the IP routing table implemented on the ingress side. The state can then go to the detection state based on the condition in Table II.

In the detection state, anomalous traffic will be detected based on the mean of queue depth. Our proposed IoT anomaly detection mechanism is implemented at the egress of the packet processing pipeline on the data plane and it is shown in the red block (9). To implement this mechanism, we utilize

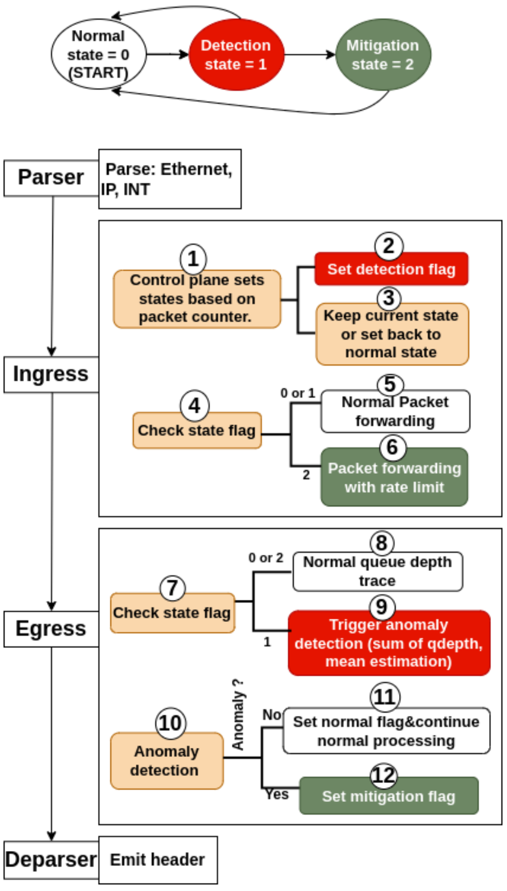


Fig. 3. IoT anomaly detection pipeline in P4.

a stateful register in P4 for storage, in which the queue depth value on the output interface is captured at the moment of transmission of probe packets. For instance, if 128 probe packets are transmitted, we record 128 queue depth values in the storage. After accumulating a certain number of queue depth values, we compute their mean value using the bitshift operator since the modulo and division operators are not available in the P4 language. This mean value is subsequently compared to a threshold. If it surpasses the threshold, the traffic is considered abnormal (10,12). The threshold itself was determined through statistical analysis during normal and malicious traffic scenarios in Section III. Then, the mitigation can be started if this mean value is higher than the baseline mean of the INT data under normal traffic conditions (12).

In the mitigation state, IoT attacks are mitigated by limiting the rate of the packet's incoming interface. To achieve this, rate limiting is implemented using the meter object of P4 at the ingress part of the pipeline on the data plane, as shown in the green block (6) of Fig. 3. P4 supports two types of meters: Indirect and Direct meters. In this implementation, we utilized indirect meters, which can be addressed by index [19]. To configure the rate limit parameters, we set up the corresponding traffic parameters in the control plane. The

TABLE II
STATE TRANSITIONING

	Start	**Stop**
Normal	Program starts in this state. Data plane set it from the detection state (10,11).	The control plane will switch to Detection (1,2).
Detection	The control plane decides to transition to this state based on packet counters implemented in the data (1,2).	The data plane decides whether to transition to a normal (10, 11) or mitigation state based on queue depth (10,12).
Mitigation	The data plane configures this state based on the detection result (10,12).	Based on the packet counter, the control plane awakens the normal state (1,3).

BMv2 software switch utilizes two-rate three-color meters[5], so we specify the Peak Information Rate (PIR) with Peak Burst Size and the Committed Information Rate (CIR) with Committed Burst Size. The mitigation mechanism may also employ other methods, such as reflecting anomalous traffic to the source port. The control plane decides when to switch back to the normal state based on the packet counter, following the criteria outlined in Table II.

Overall, our proposed P4-based anomaly detection and mitigation approach enhances the security and resilience of IoT networks by providing real-time monitoring, detection, and countermeasures against malicious activities. Its flexibility and customization capabilities empower network administrators to tailor the system to their specific requirements, making it a valuable addition to IoT network security measures.

V. Performance evaluation

The number of queue depths used to calculate the mean value is a configurable parameter for the detection mechanism of our proposed model. We experimented with different values for this parameter, ranging from 16 to 256, to determine an optimal value that would result in high detection accuracy. Fig. 4 illustrates the relationship between the number of queue depths and the detection accuracy, helping us identify the value that provides the best performance.

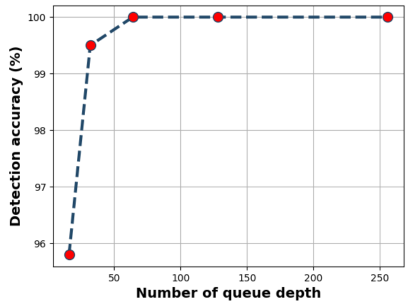


Fig. 4. Detection accuracy.

[5] https://www.rfc-editor.org/rfc/rfc2698

The average anomaly detection time is determined by multiplying the processing time (delay) of a single probe packet by the number of probe packets. A smaller number of probe packets results in faster anomaly detection.

In a real network, the packet processing delay depends on the network protocol, computational power at a node, and the efficiency of network interface cards. However, modern high-speed devices are capable of processing packets almost at line speed. In our case, probe packets can also be sent at line speed, but to clearly see and easily calculate the detection delay, we send 1 probe packet every 1 millisecond. Therefore, the anomaly detection delay is 16 milliseconds if the number of collected queue depth is 16, 32-millisecond if it is 32, and so on.

We have found that the intersection points with the highest accuracy and the acceptable delay happen when the number of queue depths is 64. With this optimal number, our proposed approach's detection delay is four times lower than the results of previous research on DDoS [3], and the detection accuracy is also higher. Additionally, it is evident that the detection delay can be absolutely low in real networks with line speeds such as 10Mbps, 100Mbps, and so on.

VI. Conclusions and future work

Our research represents the first experimental solution that employs INT data for detecting IoT attacks on the data plane (in-network). One of the primary advantages of our proposed approach is its lower detection delay, as it is directly implemented in the packet processing path. Additionally, our method leverages real-time INT data, resulting in more efficient and accurate detection capabilities. Compared to previous research, our approach exhibits relatively low detection delays and high accuracy.

Furthermore, by offering real-time monitoring, detection, and countermeasures against malicious activities, our solution provides a valuable tool to secure IoT environments. Its inherent flexibility and customization options enable network administrators to tailor it according to their specific needs, making it a valuable addition to IoT network security measures.

Despite our successful results, we acknowledge the limitations of our work. Our test environment only allowed us to test DDoS attacks with ICMP floods. To further validate our proposed approach, we plan to conduct tests on real hardware with various types of attacks. Moreover, we are aware that some IoT sensors may periodically generate large amounts of data, which we have not yet considered in our approach. Addressing such exceptional scenarios will be a valuable aspect of our future work.

.

Acknowledgment

The authors would like to acknowledge the financial support provided by project no. FK_21 138949 from the National Research Development and Innovation Fund of Hungary, as well as the contribution from Ericsson, which greatly aided the completion of this research paper.

References

[1] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, “DDoS in the IoT: Mirai and other botnets,” Computer, vol. 50, no. 7, pp. 80–84, 2017. DOI: 10.1109/MC.2017.201

[2] E. Bertino and N. Islam, “Botnets and Internet of Things security,” Computer, vol. 50, pp. 76–79, 2017. DOI: 10.1109/MC.2017.62

[3] A. D. S. Ilha, A. C. Lapolli, J. A. Marques, and L. P. Gaspary, “Euclid: A Fully In-Network, P4-Based Approach for Real-Time DDoS Attack Detection and Mitigation,” IEEE Transactions on Network and Service Management, vol. 18, no. 3, pp. 3121–3139, 2021. DOI: 10.1109/TNSM.2020.3048265

[4] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem, “Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches,” Internet of Things, vol. 7, p. 100059, 2019. DOI: 10.1016/j.iot.2019.100059

[5] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, “N-BAIoT—network-based detection of IoT botnet attacks using deep autoencoders,” IEEE Pervasive Computing, vol. 17, no. 3, pp. 12–22, 2018. DOI: 10.1109/MPRV.2018.03367731

[6] V. Timčenko and S. Gajin, “Machine learning based network anomaly detection for IoT environments,” in ICIST-2018 conference, 2018.

[7] S. S. Bhunia and M. Gurusamy, “Dynamic attack detection and mitigation in IoT using SDN,” in 2017 27th International telecommunication networks and applications conference (ITNAC). IEEE, pp. 1–6, 2017. DOI: 10.1109/ATNAC.2017.8215418

[8] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, “Detecting volumetric attacks on IoT devices via SDN-based monitoring of mud activity,” in Proceedings of the ACM Symposium on SDN Research, pp. 36–48, 2019. DOI: 10.1145/3314148.3314352

[9] E. Tsogbaatar, M. H. Bhuyan, Y. Taenaka, D. Fall, Kh. Gonchigsumlaa, E. Elmroth, and Y. Kadobayashi, “DeL-IoT: A deep ensemble learning approach to uncover anomalies in IoT,” Internet of Things, vol. 14, no. March, p. 100391, 2021. DOI: 10.1016/j.iot.2021.100391

[10] G. Altangerel and M. Tejfel, and E. Tsogbaatar, “A 1D CNN-based model for IoT anomaly detection using INT data,” in 2022 IEEE 16th International Scientific Conference on Informatics (Informatics). IEEE, pp. 106–113, 2022. DOI: 10.1109/Informatics57926.2022.10083469

[11] Dubem Ezeh, and J. de Oliveira, "An SDN controller-based framework for anomaly detection using a GAN ensemble algorithm", Infocommunications Journal, Vol. XV, No 2,pp. 29–36, June 2023. DOI: 10.36244/ICJ.2023.2.5

[12] G. Altangerel and M. Tejfel, “Study on emerging applications on data plane and optimization possibilities,” in International Journal of Distributed and Parallel systems (IJDPS) Vol 13, No. 1, January 2022. DOI: 10.5121/ijdps.2022.13101

[13] L. Sándor, G. Csaba, J. Pető and P. Vörös and G. Szabó "In-Network Velocity Control of Industrial Robot Arms." In 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22), pp. 995–1009. 2022.

[14] A. Sanghi, K. P. Kadiyala, P. Tammana, and S. Joshi, “Anomaly Detection in Data Plane Systems using Packet Execution Paths,” in SPIN 2021 - Proceedings of the 2021 ACM SIGCOMM Workshop on Secure Programmable network INfrastructure, no. 1, pp. 9–15, 2021. DOI: 10.1145/3472873.3472880

[15] Q. Kang, J. Xing, and A. Chen, “Automated attack discovery in data plane systems,” 12th USENIX Workshop on Cyber Security Experimentation and Test, CSE T 2019, co-located with USENIX Security, 2019.

[16] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, L. J. Wobker, and B. Networks, “In-band Network Telemetry via Programmable Dataplanes,” Sosr, no. Figure 2, pp. 2–3, 2015. Available: https://www.cs.princeton.edu/~nkatta/papers/int-demo.pdf

[17] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” IEEE Communications Surveys Tutorials, vol. 16, no. 3, pp. 1617–1634, 2014. DOI: 10.1109/SURV.2014.012214.00180

[18] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, “P4: Programming protocol-independent packet processors,” Computer Communication Review, vol. 44, no. 3, pp. 87–95, 2014. DOI: 10.1145/2656877.2656890

[19] L. Vanbever, “Lecture Notes - Advanced Topics in Communication Networks Programming Network Data Planes,” 2019.

Máté Tejfel received his B.Sc., M.Sc. and Ph.D. Degree in Computer Science, from ELTE, Budapest Hungary. He is currently working as an Associate Professor in the Department of Programming Languages and Compilers, Eötvös Loránd University (ELTE). His research interest includes programming languages, correctness check, SDNs, and network optimization. For more information, visit his database at 0000-0001-8982-1398 orchid-id.

Gereltsetseg Altangerel received her B.Sc. M.Sc. Degree in Information Technology, from the Mongolian University of Science and Technology (MUST). Currently, she is an assistant teacher and a Ph.D. candidate in the Department of Programming Languages and Compilers, Eötvös Loránd University (ELTE) under the supervision of Professor Tejfel Máté. Her research interest includes SDNs, deeply programmable networks, and network optimization.

For more information, visit her database at 0000-0002-1594-8158 orchid-id.