Is AI a Trick or T(h)reat for Securing Programmable Data Planes?

Bardhi, Enkeleda; Conti, Mauro; Lazzeretti, Riccardo

DOI
10.1109/MNET.2024.3451330

Publication date
2024

Document Version
Final published version

Published in
IEEE Network

Is AI a *Trick or T(h)reat* for Securing Programmable Data Planes?

Enkeleda Bardhi, Mauro Conti, and Riccardo Lazzeretti

Abstract

In the Internet of Things era, the Internet demands extremely high-speed communication and data transformation. To this end, the tactile Internet has been proposed as a medium that provides the sense of touch ability, facilitating data transferability with extra-low latency in various applications ranging from industry, robotics, and healthcare to road traffic, education, and culture. Here, programmable networks are role players in approaching the tactile Internet's low latency (≈ 1ms) pillar. Several functionalities – including security – are offloaded onto the network core employing programmable in-network pipelines. From the security perspective, Artificial Intelligence (AI) is another role player that enables the line-rate inference on the core network without involving the control plane. However, integrating AI-based security solutions in programmable devices is challenging mainly because of their constrained anatomy. Furthermore, such solutions inherit well-known adversarial AI vulnerabilities, representing an additional threat to programmable networks. Considering the above, this article discusses AI-based security solutions in programmable networks, focusing on the explored modalities of integrating AI models in programmable constrained network devices. Moreover, we elaborate on the challenges and risks of relying on AI for such mechanisms. Lastly, the article brings a visionary glimpse for future trends in this regard, raising some essential questions on the indispensability of AI for security functionalities and providing some alternative solutions.

Introduction

Tactile Internet is an evolutionary perception of communication networks that enables the real-time interaction between human beings and cyber-physical systems – i.e., autonomous automobiles, augmented reality, smart grids, industrial control, robotics, and healthcare – as if they were near. With the tactile Internet, humans and things can exchange voluminous triple-play data – i.e., audio, video, and text – as well as haptic control messages that enable the sense of touch communication. For example, consider a teleoperation application where a surgeon located far from the patient operates following not only the perception received from video and audio but also enriching the touching sense experience. Extremely low latency and high availability, reliability, and security are among the pillars expected to build the tactile Internet infrastructure. Conversely from today's systems, securing the communication for the tactile Internet systems requires additional effort to provide securely transmitted data with very low end-to-end latency. Therefore, security mechanisms against attackers should be carefully designed in the underlying network devices rather than at higher protocol layers.

Satisfying the requirements mentioned above is challenging for the underlying networks, which: *(i)* are designed to support fixed operations given their rigid framework, *(ii)* use high bandwidth by leveraging larger buffing queues for the packets, and *(iii)* mainly accommodate latency-tolerant applications. To tackle such fundamental communication and computation challenges, the research community recently leverages novel emerging yet mature technologies, including Software Defined Networking (SDN) and stateful programmable data planes. SDN decouples the control from the data plane for a more dynamic and less costly network structure, enabling the network operator to flexibly configure, manage, and control the network through application programming interfaces. Here, the control plane ensures the appropriate decisions to handle the traffic, while the data plane forwards the traffic following the rules indicated by the former. To avoid the frequent and unnecessary interactions between the controller and the data plane, the SDN paradigm evolved towards the stateful programmable data planes. Here, states and operation logic are offloaded from the controller to the data plane, accelerating the packet processing and, thus, avoiding added overhead into the network [1]. Given the flexibility, reduced latency, and fast reaction time, stateful data planes have been leveraged in recent years to enable the sense of touch for communication among different cyber-physical systems [2].

Artificial Intelligence (AI) is another promoter of tactile communication in the programmable data planes. Here, the application of AI enables intelligent mechanisms for flexible and accurate network management, including resource allocation, congestion control, routing, load balancing, fault detection, Quality of Service (QoS), and Quality of Experience (QoE). Additionally, AI is leveraged to improve network security, ranging in different applications of intrusion detection and traffic classification mechanisms on real-time at

Digital Object Identifier: 10.1109/MNET.2024.3451330
Date of Current Version: 18 November 2024
Date of Publication: 29 August 2024

Enkeleda Bardhi (corresponding author) and Riccardo Lazzeretti are with the Department of Computer, Control and Management Engineering, Sapienza University of Rome, 00184 Rome, Italy; Mauro Conti is with the Department of Mathematics, University of Padua, 35122 Padova, Italy, and also with the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628 CD Delft, The Netherlands.

line rate for fast detection and mitigation of security threats [3], [4], [5], [6], [7], [8].

Despite the benefits of engaging AI for in-network security solutions in programmable networks, various questions regarding its feasibility and advantages are raised. Foremost, the application of AI to such networks is bounded to the network devices' – i.e., switches and Network Interface Card (NIC) – available computation and memory resources that limit the means for embedding the algorithms into such devices. Alternatively, dedicated hardware – e.g., Field Programmable

Gate Array (FPGA) – in a collection device – e.g., border routers – is used to deploy in-network AI algorithms. Although more effective and robust, introducing dedicated hardware to the network devices leads to added computation overhead while increasing communication latency. Additionally, wisely selecting the critical collection devices that provide more complex computations, thus where the AI algorithm resides, introduces the single point of failure issue. Lastly, although massively used in heterogeneous domains, AI-based mechanisms are known to have critical unsolved issues that perturb their behavior and expose the networks to new threats. Among these, poisoning and evasion attacks weaken the model performance by modifying the data or its labels during the training and testing phase. Furthermore, the AI-based security tools are exploited to launch privacy attacks. Here, the adversary probes the model to learn more about the training data and gain information that could compromise the users' privacy.

This article elaborates on the various methods the research community explores to enable the deployment of AI models in programmable devices. Additionally, we discuss potential challenges posed for the practical deployment of AI-based security solutions on programmable networks while shedding light on the security risks that AI exposes the networks to. Lastly, we take a further step ahead by analyzing and envisioning whether the programmable networks truly need AI solutions. To this end, we propose a lightweight solution based on symbolic knowledge extraction from state-of-the-art AI models, depending on the indispensability of using AI. Lastly, we preliminary evaluate our vision accuracy compared to underlying baselines. The remainder of this article is organized as follows. In the section "AI-Based Security for Programmable Networks," we broadly describe the classes of state-of-the-art AI-based security solutions in programmable networks. After that, in the section "AI Solutions in Programmable Networks: Challenges and Security Risks," we elaborate on the challenges of deploying AI solutions in programmable network devices – i.e., switches and NIC – while listing the security risks due to such deployment. Lastly, we provide an overall discussion and our vision in the section "Our Vision" and conclude in the section "Conclusion."

AI-Based Security for Programmable Networks

Programmable network devices have been the breakthrough for making network operators less dependent on device vendors. Here, the flexible programming network devices allow the network operator to delegate the packet processing and forwarding logic to the network devices on the data plane while being able to promptly and regularly reconfigure them. In the following, we briefly describe the programmability of network devices in the section "Programmable Network Devices" and after that elaborate on the AI-based in-network security mechanisms in the section "AI-Based In-Network Security Mechanisms."

Despite the benefits of engaging AI for in-network security solutions in programmable networks, various questions regarding its feasibility and advantages are raised.

Programmable Network Devices

In the latest years, the Software Defined Networking (SDN) paradigm has changed the way of networking by decoupling the control plane from the data plane. Here, a centralized controller in the control plane is in charge of determining the packet processing policies—e.g., forwarding rules, packet dropping, header rewriting, and device operation. In turn, the data plane devices execute the injected policies from the control plane, thus satisfying the high-performance requirements for the network. However, the breakthrough brought by SDN pushed the demand for a more flexible and dynamic data plane rather than relying on fixed-operation devices that depend on the underlying hardware and software. The *programmable data plane* concept is introduced to overcome such an issue. Here, a programmable data plane refers to the network devices that enable programmable, flexible, dynamic, and quick reconfiguration for the packet processing functionalities from the network control plane. In the class of programmable devices, programmable switches and Network Interface Card (NIC) [1] are the essential components.

Among the basic functional operations, packet processing encompasses *parsing, classification, modification, deparsing, and forwarding*. The current programmable network device architectures originated from Protocol-Independent Switch Architecture (PISA) and their pipeline is developed based on the Reconfigurable Match-action Tables (RMT) model. A single pipeline encompasses three building blocks – i.e., parser, deparser, and match-action pipeline – as shown in Figure 1. PISA is the build base for a considerable amount of open-source and commercial implementations, such as Intel Flexpipe, Barefoot Tofino, v1model, Programmable NIC Architecture (PNA) Portable Switch Architecture (PSA) and SimpleSumeSwitch. Here, most of the switches' architectures include two main pipelines—i.e., ingress and egress pipelines. Generally, the packet initially enters a programmable parser that dissects the packet into a sequence of fields—i.e., Packet Header Vector (PHV). After that, the matching stages match the extracted packet headers while the action stages engage Arithmetic Logical Units (ALU) for applying actions – e.g., modifications and calculations – on the headers. Lastly, the deparser recomposes the packet by serializing the PHV with packet payload and sends the packet towards another pipeline or an interface.

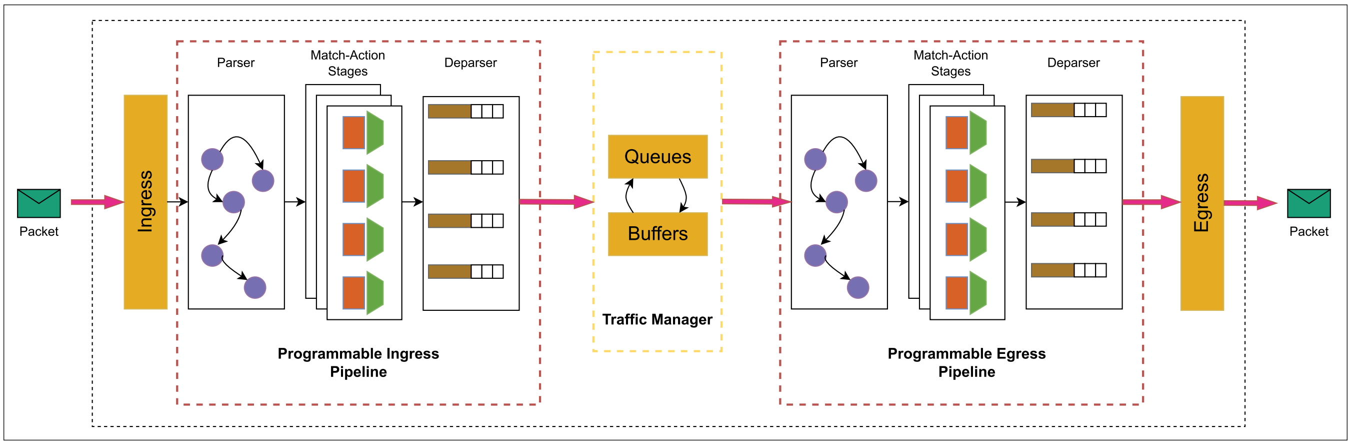

FIGURE 1. The workflow for an RMT-switching architecture comprises the ingress and egress pipelines. Each packet flowing on the pipelines traverses a parser, match-action stages, and a deparser.

To date, AI in the programmable data planes covers various security tasks, including traffic and node classification, intrusion and anomaly detection, and Denial of Service (DoS) and Distributed Denial of Service (DDoS) attack detection.

Although offering several benefits for line-rate packet processing, programmable network devices are bound to memory and computing limitations. From the memory perspective, such devices are equipped with small-sized and very-fast memory – i.e., Ternary Content-Addressable Memory (TCAM) or Static Random Access Memory (SRAM) – that enables high-speed packet processing. However, the restriction in the memory size narrows the number of applications that can be implemented in the data plane. For example, security and load balancing applications ask for memory-based network traffic diagnosis actions, including per-flow or per-packet parsing, monitoring, and state maintenance. Therefore, such applications must trade-off between memory requirements and performance. Solutions that increase the memory size or add an external memory have been proposed to overcome the memory limitation. Although valid, the former increases the consumption of the chip area, which in turn affects the packet processing speed. Instead, the latter introduces extra latency and, thus, makes the programmable data planes unsuitable for tactile applications. Lastly, the computing limitations of state-of-the-art devices – e.g., 12 stages per pipeline and two to four pipelines per device in Barefoot Tofino switches – restrict the implementation of network functionalities in the data plane. Furthermore, the programmable devices are bound to basic operations – e.g., addition and subtraction – limiting the type of data plane applications and their extent. Applying AI-based security mechanisms that require complex operations – e.g., multiplication, division – becomes challenging and bound to the devices' hardware design.

AI-Based In-Network Security Mechanisms

In the last decade, Machine Learning (ML) and, more generally, Artificial Intelligence (AI) algorithms have been widely explored for network security tasks. Here, AI facilitates the construction of security systems by increasing resilience and the ability to detect multiple vulnerabilities. Additionally, intelligent security systems are easily adaptable to the environment and vulnerability pattern changes. Only recently, academia and industry are investigating security solutions that engage AI algorithms in the programmable devices – i.e., switches and smartNICs – located on the data plane.

To date, AI in the programmable data planes covers various security tasks, including traffic and node classification, intrusion and anomaly detection, and Denial of Service (DoS) and Distributed Denial of Service (DDoS) attack detection. As shown in Figure 2, the programmable data planes can be leveraged to execute different stages of the AI-based security mechanisms—i.e., feature extraction and processing, model training and deployment, results and the corresponding response. In particular, the feature extraction and processing stage extracts relevant features from incoming packets. Generally, such features are mainly collected on a per-packet basis—e.g., the protocol type, the packet length, or port number. Conversely, there exist also a class of features that can be extracted on a per-flow basis. That is, data from a set of packets on a communication flow are examined for calculating the desired features—e.g., number of packets, the variance of packet length, or flow duration [3]. In a programmable device, the per-packet feature extraction process is accommodated in the parser module of the ingress pipeline, which extracts the features from the packet headers. Instead, per-flow features that correspond to an aggregation operation – e.g., count – can be represented via predetermine update rules on match-action tables, while summary operations – e.g., maximum, minimum, variance – are more complex and still achievable in the programmable devices pipeline [3]. Instead, the model training stage is primarily integrated into a central node in the control plane since it requires more resources to train complex models. In general, deploying the trained models in the network is bound to the availability of the underlying hardware architecture—e.g., type of tables or memory resources. To this end, an additional external device or the default match-action tables

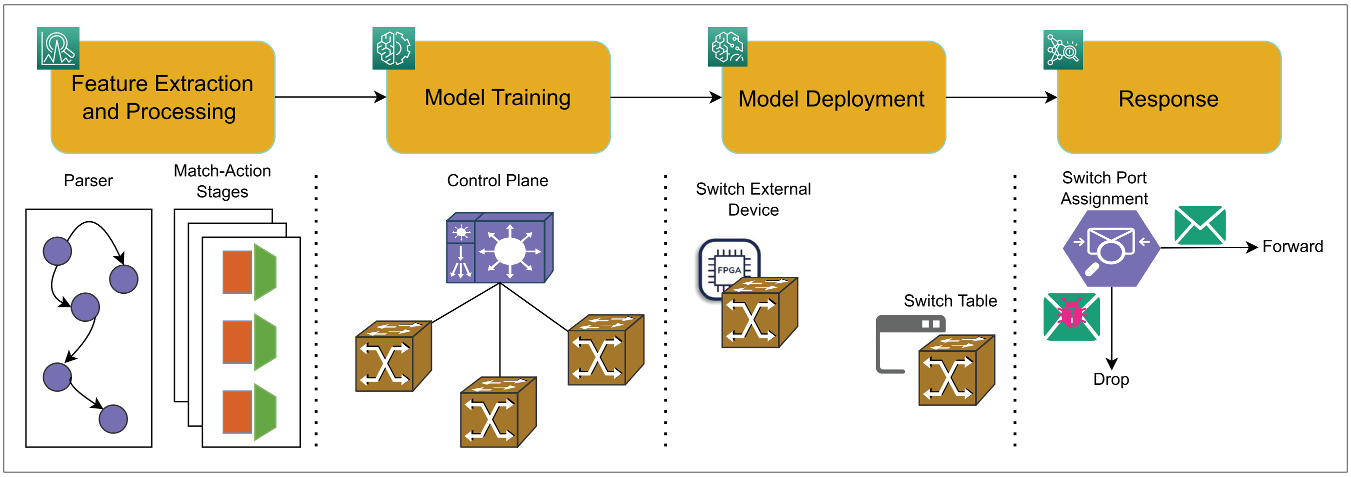

FIGURE 2. Generally, an intelligent security mechanism encompasses four stages: feature extraction and processing, model training, model deployment, and response stage. Programmable network devices can be leveraged to support these stages – e.g., the parser and match-action tables for the former stage; the computation power and resource availability of the controller can be leveraged to train the AI model; powerful external devices or the switches tables can enable the model deployment; and lastly, after the model inference, the switch follows the decision rules rather forward or drop the packet.

on the switch accommodate the trained model. Lastly, upon an inference on the incoming packet, a response is activated accordingly. In programmable network devices, such stage corresponds to the switch port assignment—i.e., if the packet is malicious, it is dropped. Otherwise, it is forwarded to the assigned port.

The AI algorithms for security mechanisms can be deployed in the programmable data planes following one of the three fashions: *(i) leveraging available resources* → commonly, the switch lookup match-action tables or registers, *(ii) memory saving* → lightweight algorithm or alternative memory saving data structures, and *(iii) changing the hardware design* → add supplementary hardware or change the underlying hardware design. Table 1 summarizes the state-of-the-art AI-based security mechanisms for three security tasks—i.e., DDoS detection, traffic classification, and anomaly detection. For the straightforward *(i)* technique, the ML and AI algorithms are mainly represented as match-action tables [3], [4], [5], [6], where each table performs a simple operation while complex operations are usually distributed into multiple steps. In this fashion, some solutions also use the registers to count and store statistical feature values that are eventually adopted for detection and defense mechanisms [7]. Nevertheless, such a technique presents limitations regarding the type of operations and their respective complexity, mainly due to the required memory consumption. Instead, the *(ii)* technique mainly compresses the underlying algorithms to make them suitable for constrained programmable devices. The Binary Neural Networks (BNNs) are mainly used since they are proposed for the constrained battery-powered edge devices [8]. Another explored method encompasses the Neural Network (NN) into the programmable devices following a distributed fashion [9]. In particular, the neurons of the NN layers are allocated in multiple devices and coordinated for a complete operation. Additionally, to ease NN deployment, some proposals engage new data

Security Task	Reference	AI model Deployment	Approach	AI Model
DDoS Detection	[5]	(i) Available Resources	Match-action Tables	RF, SVM, KNN, RNN
	[6]	(i) Available Resources	Match-action Tables	RF, KNN, SVM, NN
	[7]	(i) Available Resources	Count-sketch data structure	Entropy
Traffic Classification	[4]	(i) Available Resources	Match-action Tables	DT, SVM, NB, K-Means
	[3]	(i) Available Resources	Match-action Tables	DT, RF
	[10]	(ii) Memory Saving	Flow Marker Accumulator data structure	MNB, XGBoost, RF
Anomaly Detection	[8]	(ii) Memory Saving	Lightweight BNN with bitwise logic functions	Binary NN
	[9]	(ii) Memory Saving	Distribute NN neurons	NN
	[11]	(iii) New Hardware Design	Attach an accelerator	NB, NVB
	[12]	(iii) New Hardware Design	Complex operations with map-reduce	SVM, DT, RF, NN, K-Means

TABLE 1. AI-based Security Mechanisms. Legend: RF-Random Forests, DT-Decision Tree, KNN-K Nearest Neighbors, SVM-Support Vector Machine, NN-Neural Network, RNN-Recurrent Neural Network, NB-Naive Bayes, MNB-Multinomial Naive Bayes, and NVB-Naive Variational Bayes.

structures – e.g., flow accumulator [10] or count sketch – to optimize the memory allocation. All the above attempts yield scarce performance compared to *(i)* technique since the models are simplified and, thus, lose accuracy. Lastly, the *(iii)* technique bypasses the limitations introduced by the programmable devices by equipping them with additional hardware [11] – e.g., accelerators – that enable the use of complex AI algorithms for security tasks. Alternatively, new hardware is added in the device pipeline where the AI algorithms [12] are offloaded. Although valid, adding accelerated devices in the network infrastructure introduces a relevant deployment and maintenance cost for the network operator as it requires specialized skills and vendor assistance. Since the goal of this article is to investigate the feasibility of AI in non-accelerated programmable network devices, we consider only commodity ASIC switches and smartNICs for further investigation.

AI Solutions in Programmable Networks: Challenges and Security Risks

The most significant challenges identified for deploying AI-based security mechanisms in programmable data plane devices include the quarrel between the AI models' complexity and the constrained nature of such devices. *Here, we emphasize that our analysis takes into consideration two prototypes of network devices—i.e., Barefoot Tofino switches and Netronme smartNICs*. Indeed, these two prototypes represent the mostly explored network devices for in-network AI deployment, since they are fundamental components in the network that perform packet processing and recently, are increasingly embodying multiple programmable features. Therefore, we here investigate if and to what extent these network devices can perform as an accelerator. While most of the ML and AI models require complex operations and considerable memory resources, the above-mentioned programmable devices only support simple arithmetic operations – e.g., addition, subtraction, shift – and logical or relational operations in few operation steps. Instead, in terms of memory the smartNIC Netronome NFP4000 offers a 4MB SRAM for the Internal Memory Unit (IMU) while the Barefoot Tofino switch enables a 100MB SRAM.

On the other hand, deploying the primarily used algorithms for inference – i.e., Decision Tree (DT), Support Vector Machine (SVM) and NN – demands floating-point support, while for some of them – i.e., SVM and NN – requires multiplication, division and matrix operations [1]. Generally, the need for more support for such operations limits the type of algorithms and their application scenarios in the considered data plane devices. Furthermore, such limitation is commonly translated into a trade-off situation—i.e., the accuracy and performance are sacrificed to adapt AI models on the data plane devices. From the AI perspective, the feature extraction phase requires per-packet or per-flow examination at line rate for obtaining the required traffic features. Additionally, the feature processing engages additional operations on the obtained features. For example, calculating the per-flow packet distribution requires calculations on the data observed in the specified time window. All the operations mentioned above require state preservation on the underlying programmable data plane devices, translating into additional overhead and increasing communication and processing latency. Additionally, the memory size affects the number of states and flows that can be stored and processed in the data plane devices.

From the security and privacy perspective, using the AI models for network security tasks presents several peculiarities. Generally, delegating the detection and prevention security tasks to the switching devices opens the door for flooding attacks where the switches are overloaded with invalid requests and, thus, unable to provide the detection process. In addition to such issues, using AI algorithms to design data-driven mechanisms exposes the network to other issues, mainly malicious activities that compromise the model's integrity, user privacy, and system performance. One of the demonstrated shortcomings for the AI-based mechanisms relates to the model inference perturbations triggered by slightly modified input data. For example, the Generative Adversarial Networks (GAN) are broadly used to generate malicious samples that lead to classification errors by the attack detection mechanism, thus modifying the model's integrity. Such an adversarial activity leads to an incorrect network state that tricks reliability and availability. Another demonstration of the issues presented by the AI-based mechanisms concerns the implementation and deployment of the AI models in the considered programmable devices, commonly using the high-level language—i.e., Programming Protocol-independent Packet Processors (P4). Indeed, P 4 has diminished the complexity of designing and implementing the desired network applications compared to the hardware-specific languages that tend to be more prone to the error. However, being a high-level programming language, P 4 is still exposed to code-oriented vulnerabilities that the adversarial might exploit. Instead, from the training perspective, the adversaries can manipulate the AI models by leveraging the so-called backdoors, influencing the training process, and implanting the adversarial samples.

ML Algorithm	Score	Rule Extraction Algorithm	Score
MLP Classifier	84%	GridEx	80.5%
MLP Classifier	84%	CART	80.4%
RuleCOSIClassifier	98%	RuleCOSI+	97%
FCNetForHypinv	88.4%	HYPINV	71.4%
DeepRedFCNet	87%	DeepRED	78.8%
Decision Tree Classifier	90%	REfT	79.4%
Gradient Boosting Classifier	98%	TE2Rules	97.5%

TABLE 2. ML Baseline VS Rule extraction algorithm for the CSE-CIC-IDS2018 dataset.

ML Algorithm	Score	Rule Extraction Algorithm	Score
MLP Classifier	75%	CART	72.3%
RuleCOSIClassifier	98%	RuleCOSI+	95%
FCNetForHypinv	75.5%	HYPINV	63.2%
DeepRedFCNet	77.2%	DeepRED	70.2%
Decision Tree Classifier	72.6%	REfT	72%
Gradient Boosting Classifier	98%	TE2Rules	98%

TABLE 3. ML baseline VS Rule extraction algorithm for the CTU-2013 dataset.

Our Vision

The constrained nature of programmable devices can be bypassed by delegating the training process of AI models in the control plane. Indeed, existing state-of-the-art solutions that overcome the computational limitations offload the heavy calculation burden to the control plane or an external CPU. Although valid, the proposed scheme only scales in large scenarios with many devices that require updates from the control plane, thus adding communication latency. Similarly, large extensions of SRAM or TCAM can be considered for the memory constraints, which comes at a high cost for gaining some flexibility and scalability. Nevertheless, such a solution should be adequately designed to minimize the scheduling from and to the extended memory.

Given the current advancements in the integration of AI-based security mechanisms in the programmable network devices, we raised a fundamental question: *Is AI always preferred for securing the networks?* The answer to the above question can be positive and negative depending on the nature of the task to be solved, the dimension of the underlying network, and various constraints. Arguably, AI-based mechanisms are best conceptualized for complex tasks – e.g., image processing, text processing – where automating is paramount. Instead, for most network security tasks, AI can be used alternatively. However, that is not always the case. For example, consider an industrial cyber-physical system composed of many devices – e.g., IoT sensors – deployed in a distributed fashion. In such a scenario, following a data-driven approach to building a security mechanism – e.g., an anomaly detection system – would be reasonable since its primary purpose is to scale the detection in all the networks composed of distributed devices. In the context of accommodating the AI-based security mechanisms in the programmable network devices – e.g., switches and smartNICs –, the answer for most of the cases is: *programmable devices do not really dream of AI [4]*. And, to make matters worse, reaching the 1 ms round-trip-time communication latency requirement for tactile Internet leveraging the programmable network devices becomes beyond the bounds of possibility. Recall the previous example of the industrial cyber-physical system. According to [13], running the AlexNet model for inference requires an overall time of 96*ms* on the single sample, which is beyond the tactile Internet latency requirements. In a more real implementation on a smartNIC, [8] reports the feature extraction and inference for running a three layer BNN requires 50μs. Therefore, using AI models in such a scenario would reach the 1ms latency in less than 20 communication hops. Additionally, such a latency number is achieved thanks to the use of BNN models, that sacrifice the accuracy – i.e., 80% in a binary classification task – for making the model more lightweight. In this perspective, another question is: *Is it needed to deploy the AI model in each hop on the network?* To provide an answer, more research should be placed to study where to place the AI-based security mechanism in order to reduce the network latency.

Given the considerations mentioned above, our vision for the future of the integration of AI in the programmable devices for network security tasks branches in two directions:

- *Is AI an Indispensable Tool? → Make it simple!* To deal with the constraints regarding the supported computational operations and memory, the programmable devices would better prefer more lightweight solutions while also providing good performance and low overhead. To this end, we envision the adoption of explainable AI (xAI) techniques to interpret the model's rules correctly. In particular, the symbolic and sub-symbolic knowledge extraction from the trained AI models [14] enables the rule extraction that can be combined with expert knowledge for providing a more robust security mechanism. Here, the extracted information from the models is mainly expressed as logic rules that can be represented as a concatenation of feature conditions in the pipeline of the programmable devices. **Such simple logic rules cannot capture the complexity of the original model and provide the same accuracy.** However, representing such simple rules requires less computation and memory overhead than the required complexity to represent the model itself. For a better understanding, consider a Random Forest (RF) model, with N trees trained using M features for a DDoS detection mechanism using the detection dataset. Most state-of-the-art techniques represent such a model by creating M match-action tables and N processing pipelines on the switch. Conversely, using the symbolic rule extractor on the trained model would yield logic rules of the following form: *dataset(PacketLength, SrcIP, TCPSrcPort, TCPDstPort, UDPSrcPort, UDPDstPort, ATTACK):- PacketLength > 1.5, SrcIP in [192.168.1.103 - 192.168.4.118], TCPSrcPort in [17, 25]*, which are indeed simpler to be transferred to the network devices. Without loss of generality, the use of rules enables the common ML models and NN to be still used for security tasks, making them simple and suitable for programmable devices. To show to what extent these rules represent the baseline ML models, we first select a group of state-of-the-art rule extraction algorithms and two datasets containing per-packet and per-flow features, respectively. Then, we compare the accuracy of ML baselines and the set of rules extracted using the underlying extraction algorithms. Tables 2 and 3 depict the results. As expected, there is some acceptable accuracy drop when the symbolic rules are used instead of the ML classifier. Interestingly, some of the algorithms – e.g., RuleCOSI+ and TE2Rules – can maintain the same accuracy as the baselines, confirming our vision.
- *Is AI a Preferable but Not an Indispensable Tool? → Keep things simple!* On the other hand, if AI is not an indispensable tool to enable network security mechanisms, then simpler techniques should be used. Wolsing et al. [15] argue that industrial intrusion detection systems can perform proportionately to state-of-the-art, i.e., sufficient, independent, meaningful, portable, local, and efficient AI

In particular, we envision the use of explainable AI techniques whenever the AI-based security mechanisms are indispensable or simpler non-AI mechanisms otherwise.

based systems. All the more so, such reflection must be adopted for programmable network devices where simplicity is appreciated.

Conclusion

This article presents the security mechanisms in the programmable network devices that leverage the ML and AI algorithms. The article argues that the ML and AI pipelines suit the programmable device's pipeline. However, such devices have limited computation and memory resources, making the deployment of the learning-based models cumbersome. While presenting the intrinsic issues for adopting AI for security in programmable devices, the paper looks ahead by furnishing a novel vision of how to deal with these issues. In particular, we envision the use of explainable AI techniques whenever the AI-based security mechanisms are indispensable or simpler non-AI mechanisms otherwise. Lastly, we preliminary evaluate the proposed vision over state-of-the-art network security datasets and extraction algorithms, confirming its validity.

References

[1] W.-X. Liu et al., "Programmable data plane intelligence: Advances, opportunities, and challenges," *IEEE Netw.*, vol. 37, no. 5, pp. 122–128, Sep. 2023.
[2] A. Aijaz et al., "Realizing the tactile internet: Haptic communications over next generation 5G cellular networks," *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 82–89, Apr. 2017, doi: 10.1109/MWC.2016.1500157RP.
[3] C. Busse-Grawitz et al., "pForest: In-network inference with random forests," 2019, *arXiv:1909.05680*.
[4] Z. Xiong and N. Zilberman, "Do switches dream of machine learning?: Toward in-network classification," in *Proc. 18th ACM Workshop Hot Topics Netw.*, Nov. 2019, pp. 25–33.
[5] M. Roshani and M. Nobakht, "HybridDAD: Detecting DDoS flooding attack using machine learning with programmable switches," in *Proc. 17th Int. Conf. Availability, Rel. Secur.*, Vienna, Austria, Aug. 2022, p. 30.
[6] F. Musumeci et al., "Machine-learning-enabled DDoS attacks detection in P4 programmable networks," *J. Netw. Syst. Manage.*, vol. 30, no. 1, p. 21, Jan. 2022.
[7] Â. C. Lapolli, J. A. Marques, and L. P. Gaspary, "Offloading real-time DDoS attack detection to programmable data planes," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manag. (IM)*, Apr. 2019, pp. 19–27.
[8] G. Siracusano et al., "Re-architecting traffic analysis with neural network interface cards," in *Proc. 19th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*. Renton, WA, USA: USENIX Association, Apr. 2022, pp. 513–533.
[9] M. C. Luizelli et al., "In-network neural networks: Challenges and opportunities for innovation," *IEEE Netw.*, vol. 35, no. 6, pp. 68–74, Nov. 2021.
[10] D. Barradas et al., "FlowLens: Enabling efficient flow classification for ML-based network security applications," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–18.
[11] T. Mai et al., "In-network intelligence control: Toward a self-driving networking architecture," *IEEE Netw.*, vol. 35, no. 2, pp. 53–59, Mar. 2021.
[12] T. Swamy et al., "Taurus: A data plane architecture for per-packet ML," in *Proc. 27th ACM Int. Conf. Architect. Support Program. Lang. Oper. Syst.*, Mar. 2022, pp. 1099–1114.
[13] D. Sanvito, G. Siracusano, and R. Bifulco, "Can the network be the AI accelerator?" in *Proc. Morning Workshop Netw. Comput.*, Budapest, Hungary, Aug. 20, 2018, pp. 20–25, doi: 10.1145/3229591.3229594.
[14] A. Himmelhuber et al., "Detection, explanation and filtering of cyber attacks combining symbolic and sub-symbolic methods," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Singapore, Dec. 2022, pp. 381–388.
[15] K. Wolsing et al., "Can industrial intrusion detection be SIMPLE?" in *Proc. 27th Eur. Symp. Res. Comput. Secur. (ESORICS)*, vol. 13556, Copenhagen, Denmark. Cham, Switzerland: Springer, Sep. 2022, pp. 574–594.

Biographies

Enkeleda Bardhi (bardhi@diag.uniroma1.it) received the master's degree in ICT for internet and multimedia from the University of Padua in 2020 and the Ph.D. degree in computer science engineering from the Sapienza University of Rome in 2024. From October 2022 to February 2023, she was a Visiting Ph.D. Student with the Delft University of Technology, The Netherlands. From August 2023 to November 2023, she was a Visiting Ph.D. Student with Purdue University, USA. Her research interests include network security and privacy and the application of machine learning on network security tasks.

Mauro Conti (Fellow, IEEE) received the Ph.D. degree from the Sapienza University of Rome, Italy, in 2009. He was a Post-Doctoral Researcher with Vrije Universiteit Amsterdam, The Netherlands. In 2011, he was an Assistant Professor with the University of Padua, where he became Associate Professor in 2015 and Full Professor in 2018. He has been a Visiting Researcher with GMU, UCLA, UCI, TU Darmstadt, UF, and FIU. He is currently a Full Professor with the University of Padua, Italy. He is also affiliated with TU Delft and the University of Washington, Seattle. His research is funded by companies, including Cisco, Intel, and Huawei. His main research interests include security and privacy. In this area, he has published more than 600 papers in topmost international peer-reviewed journals and conferences. He is a fellow of AAIA and the Young Academy of Europe and a Distinguished Member of ACM. He was awarded the Marie Curie Fellowship by the European Commission in 2012 and a Fellowship by the German DAAD in 2013. He was the Program Chair of TRUST 2015, ICISS 2016, WiSec 2017, ACNS 2020, CANS 2021, CSS 2021, WiMob 2023, and ESORICS 2023; and the General Chair of SecureComm 2012, SACMAT 2013, NSS 2021, ACNS 2022, RAID 2024, and NDSS 2026 and 2027. He was an Associate Editor for several journals, including IEEE Communications Surveys and Tutorials, IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Forensics and Security, and IEEE Transactions on Network and Service Management. He is the Editor-in-Chief of IEEE Transactions on Information Forensics and Security and the Area Editor-in-Chief of IEEE Communications Surveys and Tutorials.

Riccardo Lazzeretti (Senior Member, IEEE) received the Ph.D. degree from the University of Siena, Italy, in 2012. He was a Visiting Researcher with the Philips Laboratory, Eindhoven, The Netherlands. He was a Post-Doctoral Researcher with the University of Siena and the University of Padua, Italy. He was an Assistant Professor with the Sapienza University of Rome in 2017, where he was an Associate Professor in 2022. He is currently an Associate Professor with the Sapienza University of Rome, Italy. His main research interests include security and privacy and applied cryptography. He is an Associate Editor of IEEE Transactions on Information Forensics and Security, *Journal of Information Security and Applications* (Elsevier), and *Computer Networks* (Elsevier).