Ultra-Low Latency User-Plane Cyberattack Detection in SDN-based Smart Grids

Aristide Tanyi-Jong Akem
aristide.akem@imdea.org
IMDEA Networks Institute
Universidad Carlos III de Madrid
Madrid, Spain

Michele Gucciardo
michele.gucciardo@imdea.org
IMDEA Networks Institute
Madrid, Spain

Marco Fiore
marco.fiore@imdea.org
IMDEA Networks Institute
Madrid, Spain

ABSTRACT

Modern power grids are smart, comprising millions of electronic devices interconnected by communication networks. This exposes them to a wide range of cyberattacks which could lead to power outages and data breaches with far-reaching consequences. Thus, the timely detection of such attacks is essential. Machine Learning (ML) models are widely used for cyberattack detection in Smart Grids (SG) based on Software-Defined Networks (SDN). However, these models either run in external servers or in-network, fully in the application or control plane or distributed between the control and user planes. In all three cases, the models do not run at line rate and incur hundreds of milliseconds of delay in attack detection. This paper explores how ML inference in programmable switches can enable accelerated attack detection and mitigation in SGs at line rate with sub-microsecond delay. The proposed workflow brings the concept of user plane inference to SDN-based SGs and deploys a trained Decision Tree (DT) model into the switch pipeline for real-time inference on live traffic. The model is implemented in a testbed with production-grade Intel Tofino switches, where experiments are run with a DNP3 intrusion detection dataset. Results reveal how the model can distinguish multiple attacks against SGs with an accuracy of 99%, incurring a delay within 356 nanoseconds, while consuming a tiny portion of the available resources in the switch.

CCS CONCEPTS

• **Networks** → **Programmable networks**; • **Computing methodologies** → **Machine learning**; • **Security and privacy** → **Intrusion detection systems**; • **Hardware** → **Smart grid**.

KEYWORDS

Smart grid, in-switch inference, cyberattack, machine learning, P4

ACM Reference Format:
Aristide Tanyi-Jong Akem, Michele Gucciardo, and Marco Fiore. 2024. Ultra-Low Latency User-Plane Cyberattack Detection in SDN-based Smart Grids. In *The 15th ACM International Conference on Future and Sustainable Energy Systems (E-Energy '24), June 4–7, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3632775.3661995

E-Energy '24, June 4–7, 2024, Singapore, Singapore

1 INTRODUCTION

Recent cyberattacks on smart grids (SGs) have shown how the evolution of power grids into complex smart systems with millions of interconnected components exposes them to cyberattacks. The multi-event cyberattacks on Ukrainian critical industrial control system (ICS) infrastructure with two disruptive events in October 2022 [28], and the hacking of India's power grid in 2021 [17] show how cyberattacks on SGs have become a growing security concern to the power industry and even national governments [13, 24].

Modern SGs are equipped with Supervisory Control and Data Acquisition (SCADA) systems which are cyber-physical systems for real-time monitoring and control of the electricity distribution network [27]. Traditionally, these systems ran on private networks but as they are increasingly being connected to the Internet to reduce capital and operating expenditure, they are a major loophole for attackers to exploit. The introduction of 5G and 6G which highly use Software-Defined Networking (SDN) and Network Function Virtualization (NFV) that massively employ HTTP and REST API protocols will further expose SGs to security vulnerabilities [38].

In addition, widely used SCADA system protocols like the IEC 61850 and the Distributed Network Protocol 3 (DNP3) are not designed with native security [23]. This further exposes critical infrastructure like SGs to attacks like man-in-the-middle (MITM), distributed denial of service (DDoS), masquerading, jamming and spoofing which will lead to severe consequences in case of blackouts or data thefts on the SG network [13]. Hence, the timely detection of these attacks is key to mitigating the possible damage, the reason why it has attracted much attention in the research community [21].

Cyberattack detection techniques based on machine learning (ML) are widely regarded as state-of-the-art in recent surveys [5]. In SGs based on Software-Defined Networking (SDN), ML models run in the control plane or specialized cloud servers [15]. This induces a considerable delay in detecting any security threat due to the back-and-forth communication between the user plane where the packets transit and the control plane where ML runs. The delay is in the order of tens to hundreds of milliseconds [14], which may let in the system a large volume of malicious short-lived traffic, or force the SG provider to introduce substantial latency in all communications to accommodate intrusion detection procedures.

In this paper, we propose an original approach to detect cyberattacks against SDN-based SGs that can operate at ultra-low latency of tens to hundreds of nanoseconds, hence *reducing the threat identification delay by 5–6 orders of magnitude* with respect to current standards. Our solution hinges on the idea of having the ML models run directly in the user plane, performing inference on incoming packets as they are forwarded by the network switches and without

any (time-consuming) interaction with the control plane. To this end, we leverage recently developed commercial programmable user-plane switches like Intel Tofino ASICs [19] and associated network programming languages like P4 [6]. These switches are highly constrained in terms of low available memory, limited support for mathematical operations, and the number of allowed operations per packet [32]. Such limitations, combined with inherent constraints of the P4 language like the absence of loops, make deploying trained ML models in the user plane a significant challenge [45].

To date, several works have explored techniques to deploy ML models in programmable user planes, mainly using Decision Tree (DT) and Random Forest (RF) architectures due to their simple logical structure and limited operations during inference, which are amenable to implementation in the switch pipeline [2, 4, 8, 40, 46, 47]. However, no prior study has investigated the applicability or performance of user-plane inference in the context of SGs.

Our work closes the gap by exploring for the first time how network programmability can enable SG security at ultra-low latency. The main contributions of the paper are summarized as follows.

- A workflow is proposed which brings the concept of in-switch inference to the realm of SG systems, employing the P4 programming language to translate trained switch-specific tree-based models and port them onto the programmable switch pipeline to rapidly detect and mitigate cyberattacks on SG networks via pure user-plane inference.
- The solution is implemented as P4 software and encoded into an off-the-shelf Intel Tofino switch demonstrating its feasibility in real-world equipment. The source code is made public[1] to foster reproduction and encourage its adoption.
- An experimental evaluation of the proposed solution is conducted based on measurement data from a recent DNP3 intrusion detection dataset. Results shed light on how the model can detect 6 cyberattack types with an accuracy of 99%, with only 356 nanoseconds of latency, and consuming less than 5% of each of the scarce switch resources.

We believe that these contributions advance the state-of-the-art in cyberattack detection and mitigation in SG networks.

2 RELATED WORK

Several works have proposed to either apply ML for cyberattack detection in SGs [12, 15, 23, 41, 43] or to run ML inference in the user plane for high-speed inference [2, 4, 40, 42, 46, 47].

2.1 ML for cyberattack detection in SGs

Iqbal *et al.* [20] evaluate five different models namely DT, RF, K-Nearest Neighbours (KNN) and Logistic Regression (LR) to classify power system disturbances. They show that the RF outperforms its counterparts. Cao *et al.* [9] explore the invasion pathway of false data injection attacks (FDIA) on cyber-physical power systems and propose a focal-loss-lightgbm (FLGB) ensemble classifier to detect such attacks accurately. Siniosoglou *et al.* [33] propose an intrusion detection system named MENSA which employs a novel Autoencoder-Generative Adversarial Network (GAN) architecture for detecting operational anomalies and classifying Modbus/TCP and DNP3 cyberattacks in different SG environments. While these works show the potential of ML/DL for attack detection in SGs, they do not implement the models in the SDN architecture.

Holik *et al.* [15] propose INPS, an in-network protection system for industrial control systems. The AI module based on an externally trained neural network is lodged in the ONOS controller of the SDN architecture and enables attack detection in the network. El Houda *et al.* [16] propose BoostIDS, a system leveraging an efficient boosting feature selection algorithm and a Lightweight Boosting Algorithm (LBA) to timely and effectively detect attacks on SGs in an SDN environment. The model runs in the controller and the system is evaluated in Mininet. DIDEROT [30] proposes an intrusion detection and protection system comprising a first detection layer relying on a DT classifier which recognises DNP3 cyberattacks, and a second detection layer which uses an autoencoder DNN to detect DNP3 anomalies. DIDEROT interfaces with the SDN controller via a REST API and enables attack detection and mitigation. The above works enhance SG protection against cyberattacks by implementing the models in the control plane. Yet, they do not exploit the user plane to accelerate detection speed.

Ndonda *et al.* [25] exploit the user plane and propose a two-level intrusion detection system for SDN-based industrial control networks. The first level runs in the switch, exploiting flow and Modbus whitelists implemented in P4 for efficient real-time monitoring. The second level is a deep packet inspection engine (DPI) communicating with an SDN controller to update the whitelists of the first level. However, the solution is not ML-based and requires a DPI engine running on an external host which does not run at line rate and entails additional costs.

2.2 ML inference in the user plane

Advances in user plane programmability have given rise to several proposals to embed ML models into network equipment for high-speed inference. One of the earliest proposals is N2Net [34] which deploys Binarized Neural Networks (BNNs) in switches. Although BNNs are simpler than NNs, deploying them into switches is not practical, as a simple BNN with 2 layers exhausts the resources on a Tofino ASIC [34]. To enable NN deployment, N3IC [35] resorts to Smart Network Interface Cards (SmartNICs), which have fewer constraints. However, these NICs are deployed on hosts, and cost almost as much as switches while offering only a few ports.

Most works have focused on programmable switches with embedded Decision Tree (DT) and Random Forest (RF) models [1, 4, 7, 40, 42, 46]. The choice of these models hinges on the limited operations performed at inference and their simple logical structure that make them compatible with switch constraints. IIsy [42] proposed a DT mapping scheme for packet-level (PL) inference where features are mapped onto separate tables and then a final table summarizes the paths to leaves on the tree. This approach is extended to support RFs in Planter [46] and other works that employ this mapping or modifications of it [1, 2, 4, 47]. Mousika [40] introduces a PL classifier based on knowledge distillation. A teacher model is trained and then distilled into a single compact binary decision tree (BDT) which is then deployed in the switch. Flowrest [4] proposes a practical framework that enables flow-level (FL) inference in switches. Jewel [2] and NetBeacon [47] propose concurrent PL and FL inference using single and multiple models respectively.

[1]The code is available at https://www.github.com/nds-group/smart-grid

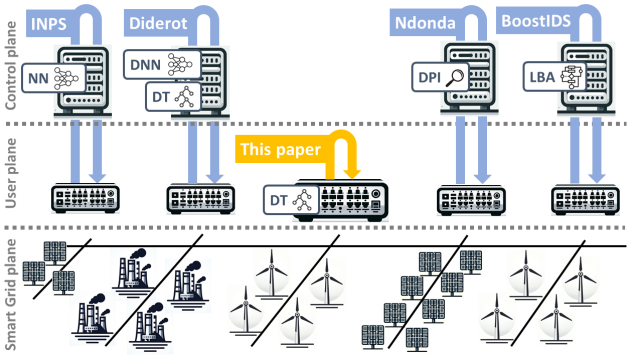


Figure 1: Overview of the workflow of the proposed solution and representative existing approaches.

The targeted use cases across existing works include device identification [1, 2, 4, 42, 46], intrusion detection and attack classification [2, 4, 40, 42, 46, 47], service or application classification [2, 4, 47], heavy-hitter detection [44], and encrypted traffic classification [3]. To the best of available knowledge, no work to date has tackled the case of cyberattack detection in SGs, neither has any of the existing works been evaluated on a relevant SG or SCADA system dataset. This paper fills the above gap by exploiting in-switch inference for cyberattack detection in SDN-based SGs.

3 IN-SWITCH CYBERATTACK DETECTION IN SDN-BASED SMART GRIDS

The cyberattack detection solution is designed as fully in-switch framework which runs entirely in programmable switches, detecting and mitigating cyberattacks on SGs without any involvement of the controller in the inference process as in most previous works as described in §2.1. The proposed workflow is described in §3.1, followed by a description of the modelling process in §3.2 and the actual in-switch implementation in §3.3.

3.1 Solution overview

An overview of the proposed solution is presented in Figure 1 alongside the workflows of closely related works like INPS [15], BoostIDS [16], DIDEROT [30] and Ndonda *et al.* [25] which all involve either the control plane or external hosts in their proposals. This paper deviates from those paradigms by performing inference entirely in user-plane switches. The SDN-based SG architecture is presented as comprising three logical planes namely *(i) the smart grid plane*; which consists of the power system network and associated devices like smart meters and power distribution units, *(ii) the user plane*; which is the communication network made up of switches and routers, and *(iii) the control plane*; which lodges SDN controllers like RYU, OpenDaylight and Floodlight which implement control functions and set user plane rules [31].

The user plane lies at the centre of this architecture, playing the critical role of transmitting sensitive data that could include consumption data, grid status, and control commands. This makes it a target for many cyberattacks against the SG network. Most existing approaches to detect and mitigate cyberattacks run in the control plane or in external systems connected to the control plane as illustrated in Figure 1, which introduces significant additional delay required for the user plane to communicate with the higher plane either to transmit features for real-time classification or to receive instructions on post-classification actions. Our solution allows for eliminating this latency by embedding ML models into the user-plane switches, so that attack detection occurs simultaneously with data transmission. This enables line rate traffic classification with extremely low latency and high throughput and reduces the reaction time to attacks by applying mitigation techniques immediately after detection.

3.2 Model preparation

Prior works have demonstrated the potential of DT and RF models for attack detection and classification in SG systems [23, 30, 36]. As these models are also the most adaptable to switch constraints as discussed in §2.2, this paper also adopts them. The switch constraints also make the model preparation phase impractical in the switch and so this phase happens entirely offline in an ML server.

Feature extraction. The model preparation phase begins with the extraction of the features from the PCAP files of the target dataset. Tshark [39] is employed to extract the header fields of packets which are used as features for per-packet inference. The 13 extracted features include packet length, source port, destination port, protocol, TCP flags (SYN, ACK, FIN, PSH, RST), TCP header length, TCP window size, UDP length, and time-to-live (TTL). Once extracted, the packets are labelled using the label information provided by the dataset and then saved in CSV files. Using only header fields ensures that data privacy is maintained as payloads are never inspected.

Model and feature selection. The feature and model selection follows a similar approach to previous works [1, 4, 8] and uses the popular Scikit-Learn Python libraries [26]. A large DT/RF model is first trained with all available features. Then the features are ranked based on their importance in the model as expressed by the Mean Decrease in Impurity (MDI). Next, the model training and evaluation are repeated, adding the features one by one, beginning with the most important, and saving the results in CSV files for later analysis. The number of trees in the RF is also used as a parameter which is limited to the range $[1, 7]$ as very large RFs will be infeasible for in-switch implementation. Unlike in previous works, the maximum depth of the DT/RF models is not limited and the trees are allowed to grow to full size, maximizing their learning abilities. In the end, the best model is selected based on a trade-off between accuracy and complexity.

Model training and mapping to M/A table entries. Once the final model and its features are selected, the model is trained and then converted to Match & Action (M/A) table entries that will be loaded onto the switch. This conversion maps the model onto the switch pipeline using the mapping scheme originally proposed in Planter [46] and re-implemented by other works [1, 4, 47]. This scheme breaks the model into *feature tables*, and *code tables*. There is a table per feature and a code table per tree. All thresholds of a given feature in the model are extracted and organized into consecutive ranges, with a code assigned per range. The combination of the range codes along the path to a given leaf constitute its codeword and these are summarized in a tree code table with as many entries

as there are leaf nodes. As there is a table per feature, the tables are independent and can all be executed in the same M/A stage of the switch pipeline. This also applies to the code tables which can all share a stage. In the case of RFs, the final classification result is obtained by a majority vote of individual tree predictions.

3.3 User-plane model implementation

The per-packet in-switch solution is designed for the Protocol Independent Switch Architecture (PISA) which has three main parts; the Parser, the Match & Action (M/A) pipeline, and the Deparser. The M/A pipeline is separated into an Ingress and an Egress pipeline by the Traffic manager. The different components of the in-switch solution are implemented into this architecture as follows.

Parsing. Packets arriving at the switch are processed by the parser which extracts their header fields and saves them in metadata. Amongst these header fields are the features required for the inference task, which are then carried by the Packet Header Vector (PHV) to the M/A pipeline where they will be used for inference.

Traffic filtering. Upon entry into the ingress pipeline, a first M/A table known as the traffic filter is applied. The purpose of this table is to match the source IP, destination IP, source port, destination port and protocol of the packet to determine if it is part of the traffic targeted for inference and which should be processed, or just background traffic which should be forwarded normally. This avoids unnecessary processing of all packets, improving efficiency.

Packet classification. If the packet is part of the target traffic, its features (header fields) are retrieved from the metadata in the PHV and then the feature tables are applied to assign codes which when concatenated will generate a code word. The tree code tables are then applied to match the generated code word to the corresponding leaf node which has an assigned class. In the case of RFs the final class is obtained as a majority vote of the tree classes.

Attack mitigation. Once the packet is assigned a class, a class-based action can be performed. In this paper, this is demonstrated by an attack mitigation M/A table which matches the class of the packet and forwards it normally if it is benign or drops it if classified as malicious. More advanced post-classification functions can be implemented depending on the needs of the SG system.

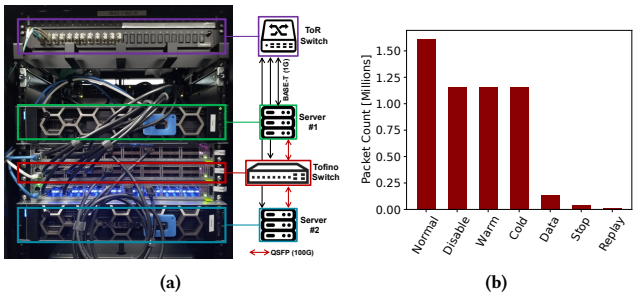
Deparsing. When the packet is classified and a forwarding action is assigned, the packet is packaged in the deparser and then sent to the egress pipeline through the traffic manager. In this work, no additional processing is done in the egress and the packet is directly forwarded or dropped depending on the action assigned by the mitigation table. Nonetheless, more processing could take place in the egress pipeline *e.g.*, applying other models for additional levels of classification [1].

4 EXPERIMENTAL EVALUATION

The solution is implemented in an Intel Tofino switch as a P4 program. Experiments are then conducted with measurement data in a hardware testbed to validate the proposed solution.

4.1 Hardware platform

The testbed is shown in Figure 2a. The components are labelled on the right of the figure. It comprises an Edgecore Wedge 100BF-QS switch, with the Intel Tofino BFN-T10-032Q chipset and 32



(a) (b)

Figure 2: (a) Setup of the experimental testbed and (b) Distribution of classes in the dataset.

100-GbE QSFP28 ports, and two DELL PowerEdge servers with Intel 8-core Xeon processors at 2GHz frequency, with 48GB of RAM, and equipped with Mellanox ConnectX-5 NICs with 100Gbps QSFP28 interfaces. In experiments, models are implemented in the switch as P4 programs for per-packet ML inference. Server #1 then injects traffic from the test pcap files to the switch through Tcpreplay [37]. Upon inference, the traffic is forwarded to Server #2, where it is captured by a Tcpdump instance and saved in a PCAP file for analysis. Moongen [11] is employed to generate and send background traffic into the switch, to provide a more realistic traffic mix. Background traffic is not inferred upon and hence does not affect classification results. A control program also runs on Server #2 and serves to configure the switch at setup and load the model to the switch in the form of M/A table entries.

4.2 Use case: DNP3 attack classification

The proposed solution is evaluated using the recent DNP3 Intrusion Detection Dataset [29] which is described in [22] and analyzed in [23]. DNP3 is a commonly used protocol in industrial control systems especially in SGs, with over 75% of North American electric utilities using it for control applications [10].

4.2.1 Dataset description. The protocol suffers from many security vulnerabilities by design and these were exploited to launch eight DNP3 attack scenarios each for a period of 4 hours from 14-19 May 2020 using opendnp3, Nmap and Scapy to generate the dataset [22]. The attacks are briefly described next.

DNP3 Slave Discovery. This scenario involved two attempts at discovering whether a given outstation uses a given IP address. It employs two scripts to send link status requests to various DNP3 links including those with the target IP address. If any outstation responds, it is identified as to be using that IP address.

DNP3 Disable Unsolicited Messages. This orders a substation to disable sending unsolicited messages to the master, implying that possible anomalies or incidents could go undetected.

DNP3 Cold Restart Message. This attack sends cold restart packets which force the outstation into a full restart which requires a self-check process, hence keeping it offline for a while.

DNP3 Warm Restart Message. This forces the outstation to restart DNP3 applications. The outstation is thus unable to respond to the master's request till the application restart process ends.

DNP3 Data Initialisation. This attack orders an outstation to reset its data to default values, hence rendering invalid all updates sent by the outstation, which will not reflect its current state.

Table 1: Classification results of the in-switch solution and the offline benchmark model.

	F1-Score		TPR		FPR		TNR		FNR	
	Offline	**In-Switch**	**Offline**	**In-Switch**	**Offline**	**In-Switch**	**Offline**	**In-Switch**	**Offline**	**In-Switch**
Normal	99.936	99.934	99.984	99.981	0.050	0.050	99.950	99.950	0.016	0.019
Disable Unsolicited	100.000	100.000	100.000	100.000	0.000	0.000	100.000	100.000	0.000	0.000
Cold Restart	100.000	100.000	100.000	100.000	0.000	0.000	100.000	100.000	0.000	0.000
Warm Restart	100.000	100.000	100.000	100.000	0.000	0.000	100.000	100.000	0.000	0.000
Data Initialization	100.000	100.000	100.000	100.000	0.000	0.000	100.000	100.000	0.000	0.000
Replay	91.162	90.900	85.489	85.360	0.005	0.006	99.995	99.994	14.511	14.640
Stop Application	100.000	100.000	100.000	100.000	0.000	0.000	100.000	100.000	0.000	0.000
Macro average	**98.728**	**98.690**	**97.925**	**97.906**	**0.008**	**0.008**	**99.992**	**99.992**	**2.075**	**2.094**
Weighted average	**99.959**	**99.958**	**99.961**	**99.959**	**0.015**	**0.015**	**99.985**	**99.985**	**0.039**	**0.041**

DNP3 Replay. This attack offsets the transmission of a request packet by a random amount of time, thereby disrupting normal DNP3 communications in the system.

DNP3 Stop Application. This forces the outstation to interrupt the DNP3 application hence making it unable to respond to DNP3 requests from the master during the attack.

4.2.2 Classification task and pre-processing. Apart from the slave discovery scenario, the dataset comes with a folder for each attack scenario, as well as three other folders for the man-in-the-middle DoS, DNP3 enumerate and DNP3 info scenarios, which are not described in [22]. The six described attack scenarios alongside the normal traffic they contain are therefore the seven classes targeted in this evaluation. The composition of the dataset in terms of the number of packets of each of the classes is shown in Figure 2b, where the imbalance between the classes is evident. This affects the classification results as shown in §5.1.

Each scenario folder has labelled CSV files from which the corresponding labels for every flow are extracted. Then using Scapy, all the PCAP files in the respective *Total PCAP Files* folders are split into train and test PCAP files, using the 75 − 25 split ratio. The features listed in §3.2 are then extracted from PCAPs and saved to labelled CSV files using Tshark and Python. The model and feature selection algorithm described in §3.2 is then executed for DT and RF model structures beginning with the full set of available features. The final selected model is the best DT which uses 4 features namely `source port`, `destination port`, `TCP data offset`, and `packet length`, and has similar scores with the best RF as will be shown in Table 1. It is selected for in-switch implementation due to its simpler nature which implies less switch resource usage.

4.3 Evaluation metrics and benchmark

4.3.1 Metrics. The classification performance is evaluated using popular metrics derived from four main measures of classification tasks, *i.e.*, true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). All the metrics are expressed as percentages to ease understanding. The metrics are explained next.

True positive rate (TPR) measures the number of positive samples that are rightly classified as positive *i.e.*, TP/(TP+FN).

False positive rate (FPR) is the fraction of negative samples incorrectly classified as positive. It is computed as FP/(FP+TN).

True negative rate (TNR) quantifies the number of negative samples that are correctly classified as negative *i.e.*, TN/(TN+FP).

False negative rate (FNR) represents the portion of positive samples that are wrongly labelled negative *i.e.*, FN/(FN+TP).

F1-score is the harmonic mean of precision (TP/(TP+FP)) and recall. It is widely used to compare overall model performances in classification tasks. It is calculated as 2TP/(2TP+FP+FN).

The metrics are computed for each class, and the final value is either a simple average of all class scores (Macro score) or a weighted average of class scores (Weighted score), using the number of samples of each class to assign weights. The macro score depicts the model performance in a scenario where all classes are of equal importance while the weighted score accounts for the imbalance in the dataset and better represents the score of any random packet in the dataset.

4.3.2 Benchmark. The performance of the in-switch solution is compared to that of a fully-grown offline RF model, running in an ML server which is oblivious to the constraints of the switch environment. This model is the best RF model obtained from the model and feature selection process and it has 3 trees and 4 features namely `source port`, `destination port`, `TCP window size`, and `packet length`. It is representative of prior work for cyberattack detection in SDN-based SG where the models run in external servers or the control plane *e.g.*, INPS [15], BoostIDS [16] and DIDEROT [30].

5 RESULTS AND DISCUSSION

The SG cyberattack detection solution is evaluated in terms of its classification performance (§5.1), consumption of scarce switch resources (§5.2), and packet processing latency (§5.3).

5.1 Classification accuracy

The classification performance of the in-switch solution and the offline benchmark in terms of the metrics presented in §4.3 are shown in Table 1. The models are evaluated based on their ability to correctly identify 7 classes which are benign traffic and 6 cyberattacks. Two experiments are conducted. In the first experiment, the model is evaluated on classifying all the attacks and normal traffic. Overall, the in-switch model correctly classifies 99.96% of all packets. Breaking this down into individual class scores, the model achieves F1-scores of between 90.90% − 100%, TPRs of between 85.36% − 100%, and FNRs of between 99.95% − 100%.

In fact, the only class where the model achieves less than 99% of any of the above scores is the Replay Message attack, where it achieves F1 and TPR of 90.90% and 85.36% respectively. This is attributed to the small number of samples of this attack in the dataset, which prevents the model from learning to predict it accurately. This effect is also perceivable in the FPR and FNR, where the model

Table 2: Usage of key switch resources.

Resource	Usage
SRAM	1.0%
TCAM	2.4%
Exact Match Input Xbar	0.6%
Ternary Match Input Xbar	3.7%
VLIW Instruction	1.8%

achieves 0.006% and 14.41% respectively, suggesting that larger training datasets or techniques to handle training data imbalances shall be used to properly detect Replay Message attacks.

For all the other attack types, the FPR and FNR are all 0% as shown in Table 1. These results shed light on how the model perfectly detects 5 of the 6 attacks, and also correctly identifies 85% of the more challenging Replay Message attack. Comparing these scores to those of the offline model also presented in Table 1, it is evident that the in-switch model scores align closely, confirming the efficiency of the model encoding in the user plane.

To evaluate the mitigation function of the solution, a second experiment is performed in which all packets belonging to any of the attack classes are automatically dropped, while only benign traffic is forwarded. At the receiver, only 0.11% of the packets are wrongly let through as benign; these packets all belong to the Replay Message class with the lowest accuracy as discussed previously. This demonstrates that the mitigation module is efficient in its role of dropping malicious packets immediately after they are classified as such, without any involvement of the controller. While ours is a simple proof of concept, any other mitigation strategy beyond packet discarding could be implemented depending on the needs of the SG system, *e.g.*, forwarding different classes of malicious traffic to honeypots for further analysis and adequate response policying.

5.2 Switch resource usage

The switch is not designed to run ML models, rather it is optimized for packet processing and forwarding. As such, deploying ML models in the switch must leave sufficient resources for other normal switch functions. The proposed in-switch inference solution's consumption of switch resources is evaluated using P4 Insight [18] which comes with Intel's Software Development Emulator (SDE) and provides a detailed analysis of compiled P4 programs and their mapping to switch resources. The resources assessed include SRAM used for exact match tables, TCAM used for ternary and range match tables, the crossbars (Xbars) used for storing the table keys, and the Very Long Instruction Words (VLIW) which carry mathematical operations.

As shown in Table 2, the model consumes less than 5% of each key switch resource. To better appreciate this resource consumption, when deployed concurrently with the standard P4 program for L2/L3 forwarding known as `switch.p4`, the cyberattack detection solution only brings an additional 7 M/A tables with a total of 227 table entries, increasing consumption of SRAM by 1.44% and TCAM by 4.37% of what the switching program already needs. Another important aspect is the number of M/A stages required for the program, which is 4 out of 12. One stage is used for the traffic filter, another for the DT model feature tables, another for the code table which assigns the classes, and the last stage for the mitigation table. This leaves many stages for more complex mitigation or post-classification procedures to be implemented based on the needs of the SG network. Overall this modest consumption of switch resources makes the solution memory efficient, allowing it to cohabit with multiple switch applications.

Table 3: Delay induced by the different inference approaches.

	Feature extraction & inference in control plane	Feature extraction in user plane & inference in control plane	Feature extraction & inference in user plane (This paper)
User-plane processing	335 ns	341 ns	356 ns
Control plane processing	1.797 ms	0.483 ms	0
Inter-plane communication	4.189 ms	1.535 ms	0
Total	**5.986 ms**	**2.018 ms**	**356 ns**

5.3 Inference latency

Finally, the key advantage of the proposed system is that inference on packets is achieved at line rate, *i.e.*, at the same speed at which the packets transit through the programmable switch. We demonstrate such a capability via measurements that are summarized in Table 3. There, we report the latency induced by our solution as the sole packet processing delay in the switch, at 356 ns: this is the time needed to both classify and forward or drop a single data packet. We compare this to two approaches: (*i*) where feature extraction and inference take place in the control plane, with the user plane sending the entire packet to its CPU via the dedicated 10 Gbps PCIe interface; (*ii*) where the user plane extracts features and sends them to the remote controller running on Server #2 for inference. We estimate the user plane and the control plane processing time respectively in nanoseconds and milliseconds. The results, shown in Table 3, are consistent with prior work on similar measurements [14] and highlight that (*i*) our solution is more than 6,000 times faster than solutions using user-plane feature extraction and control-plane inference, and (*ii*) more than 17,000 times faster than those doing both in the control plane. This confirms its superiority in terms of attack detection and response time, and sets a new standard.

6 CONCLUSIONS

Cyberattacks on SGs have become a growing concern in recent years, triggering a strong research interest in rapidly detecting and mitigating such attacks. This paper leverages ML inference in programmable switches to enable rapid detection and mitigation of cyberattacks on SDN-based SGs. A DT model is embedded into the switch and evaluated on a DNP3 attack detection and classification use case, where it achieves 99% accuracy while consuming less than 5% of each key switch resource. This is achieved while keeping packet-processing latency in the sub-microsecond range, setting a new standard. Future work will expand the scope of this work by targeting attacks on time-sensitive protocols of the IEC 61850 standard, *e.g.*, MMS or GOOSE and SV whose messages skip the network and transport layers, rendering important IP and transport layer features unavailable for inference. Overcoming this challenge will be a significant step towards detecting attacks on these protocols.

ACKNOWLEDGMENTS

A. T-J. Akem's and M. Gucciardo's research was funded by project PCI2022-133013 (ECOMOME), funded by MICIU/AEI/10.13039/501100011033 and the European Union "NextGenerationEU"/PRTR. M. Fiore's research was supported by the SNS JU and the European Union's Horizon Europe research and innovation program, under Grant Agreement No 101139270 (ORIGAMI).

REFERENCES

[1] Aristide Tanyi-Jong Akem, Beyza Bütün, Michele Gucciardo, and Marco Fiore. 2022. Henna: hierarchical machine learning inference in programmable switches. In *NativeNI 2022* (Rome, Italy). https://doi.org/10.1145/3565009.3569520

[2] Aristide Tanyi-Jong Akem, Beyza Bütün, Michele Gucciardo, and Marco Fiore. 2024. Jewel: Resource-Efficient Joint Packet and Flow Level Inference in Programmable Switches. In *IEEE INFOCOM 2024*. https://shorturl.at/djqt4

[3] Aristide Tanyi-Jong Akem, Guillaume Fraysse, and Marco Fiore. 2024. Encrypted Traffic Classification at Line Rate in Programmable Switches with Machine Learning. In *IEEE/IFIP NOMS 2024*. 1–9. https://hdl.handle.net/20.500.12761/1791

[4] Aristide Tanyi-Jong Akem, Michele Gucciardo, and Marco Fiore. 2023. Flowrest: Practical Flow-Level Inference in Programmable Switches with Random Forests. In *IEEE INFOCOM 2023*. https://doi.org/10.1109/INFOCOM53939.2023.10229100

[5] Tarek Berghout, Mohamed Benbouzid, and S. M. Muyeen. 2022. Machine learning for cybersecurity in smart grids: A comprehensive review-based study on methods, solutions, and prospects. *International Journal of Critical Infrastructure Protection* 38 (2022). https://doi.org/10.1016/J.IJCIP.2022.100547

[6] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming Protocol-Independent Packet Processors. *SIGCOMM Comput. Commun. Rev.* 44, 3 (2014).

[7] Coralie Busse-Grawitz, Roland Meier, Alexander Dietmüller, Tobias Bühler, and Laurent Vanbever. 2019. pForest: In-Network Inference with Random Forests. *CoRR* abs/1909.05680 (2019). arXiv:1909.05680

[8] Beyza Bütün, Aristide Tanyi-Jong Akem, Michele Gucciardo, and Marco Fiore. 2023. Fast Detection of Cyberattacks on the Metaverse through User-plane Inference. In *IEEE MetaCom 2023*. https://shorturl.at/dgHZ2

[9] Jie Cao, Da Wang, Zhaoyang Qu, Mingshi Cui, Pengcheng Xu, Kai Xue, and Kewei Hu. 2020. A Novel False Data Injection Attack Detection Model of the Cyber-Physical Power System. *IEEE Access* 8 (2020). https://shorturl.at/lmVW3

[10] Samuel East, Jonathan Butts, Mauricio Papa, and Sujeet Shenoi. 2009. A Taxonomy of Attacks on the DNP3 Protocol. In *Critical Infrastructure Protection III*, Charles Palmer and Sujeet Shenoi (Eds.). 67–81.

[11] Paul Emmerich, Sebastian Gallenmüller, Daniel Raumer, Florian Wohlfart, and Georg Carle. 2015. MoonGen: A Scriptable High-Speed Packet Generator. In *IMC* (Tokyo, Japan). https://doi.org/10.1145/2815675.2815692

[12] Béla Genge and Piroska Haller. 2016. A hierarchical control plane for software-defined networks-based industrial control systems. In *IFIP Networking*. https://doi.org/10.1109/IFIPNETWORKING.2016.7497208

[13] Muhammed Zekeriya Gunduz and Resul Das. 2020. Cyber-security on smart grid: Threats and potential solutions. *Computer Networks* 169 (2020). https://doi.org/10.1016/J.COMNET.2019.107094

[14] Keqiang He, Junaid Khalid, Aaron Gember-Jacobson, Sourav Das, Chaithan Prakash, Aditya Akella, Li Erran Li, and Marina Thottan. 2015. Measuring Control Plane Latency in SDN-Enabled Switches *(SOSR '15)*. ACM, Article 25. https://doi.org/10.1145/2774993.2775069

[15] Filip Holik and Petr Dolezel. 2020. Industrial Network Protection by SDN-Based IPS with AI. *Communications in Computer and Information Science* 1178 CCIS (2020). https://doi.org/10.1007/978-981-15-3380-8_17

[16] Zakaria Abou El Houda, Bouziane Brik, and Lyes Khoukhi. 2022. Ensemble Learning for Intrusion Detection in SDN-Based Zero Touch Smart Grid Systems. In *IEEE LCN*. https://doi.org/10.1109/LCN53696.2022.9843645

[17] INSIKT GROUP. 2021. China-linked Group RedEcho Targets the Indian Power Sector Amid Heightened Border Tensions. Recorded Future: https://www.recordedfuture.com/redecho-targeting-indian-power-sector.

[18] Intel. [n. d.]. P4 Insight. https://www.intel.com/content/www/us/en/products/details/network-io/intelligent-fabric-processors/p4-insight.html.

[19] Intel. [n. d.]. Tofino Switch. https://tinyurl.com/ycx79k4z

[20] Anzar Iqbal and Pooja. 2019. Intrusion detection in smart grid using machine learning approach. *Journal of Computational and Theoretical Nanoscience* 16 (2019). Issue 9. https://doi.org/10.1166/JCTN.2019.8254

[21] Julius Jow, Xiao Yang, and Wenlin Han. 2017. A survey of intrusion detection systems in smart grid. *International Journal of Sensor Networks* 23 (2017). Issue 3. https://doi.org/10.1504/IJSNET.2017.083410

[22] Vasiliki Kelli, Panagiotis Radoglou-Grammatikis, Thomas Lagkas, Evangelos K. Markakis, and Panagiotis Sarigiannidis. 2022. Risk Analysis of DNP3 Attacks. In *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*. 351–356. https://doi.org/10.1109/CSR54599.2022.9850291

[23] Vasiliki Kelli, Panagiotis Radoglou-Grammatikis, Achilleas Sesis, Thomas Lagkas, Eleftherios Fountoukidis, Emmanouil Kafetzakis, Ioannis Giannoulakis, and Panagiotis Sarigiannidis. 2022. Attacking and Defending DNP3 ICS/SCADA Systems. In *2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. 183–190. https://doi.org/10.1109/DCOSS54816.2022.00041

[24] Daisuke Mashima, Yao Chen, Muhammad M. Roomi, Subhash Lakshminarayana, and Deming Chen. 2023. Cybersecurity for Modern Smart Grid Against Emerging Threats. *Foundations and Trends® in Privacy and Security* 5, 4 (2023), 189–285. https://doi.org/10.1561/3300000035

[25] Gorby Kabasele Ndonda and Ramin Sadre. 2018. A Two-level Intrusion Detection System for Industrial Control System Networks using P4. In *5th International Symposium for ICS & SCADA Cyber Security Research 2018 (ICS-CSR)*. https://doi.org/10.14236/EWIC/ICS2018.4

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011).

[27] Dimitrios Pliatsios, Panagiotis Sarigiannidis, Thomas Lagkas, and Antonios G. Sarigiannidis. 2020. A Survey on SCADA Systems: Secure Protocols, Incidents, Threats and Tactics. *IEEE Communications Surveys & Tutorials* 22, 3 (2020). https://doi.org/10.1109/COMST.2020.2987688

[28] Ken Proska, John Wolfram, Jared Wilson, Dan Black, Keith Lunden, Daniel Kapellmann Zafra, Nathan Brubaker, Tyler McLellan, and Chris Sistrunk. 2023. Sandworm Disrupts Power in Ukraine Using a Novel Attack Against Operational Technology. Mandiant: https://www.mandiant.com/resources/blog/sandworm-disrupts-power-ukraine-operational-technology.

[29] Panagiotis Radoglou-Grammatikis, Vasiliki Kelli, Thomas Lagkas, Vasileios Argyriou, and Panagiotis Sarigiannidis. 2022. DNP3 Intrusion Detection Dataset. https://doi.org/10.21227/s7h0-b081

[30] Panagiotis Radoglou-Grammatikis, Panagiotis Sarigiannidis, George Efstathopoulos, Paris-Alexandros Karypidis, and Antonios Sarigiannidis. 2020. DIDEROT: an intrusion detection and prevention system for DNP3-based SCADA systems. In *ARES 2020* (Virtual Event, Ireland) *(ARES '20)*. Article 115, 8 pages. https://doi.org/10.1145/3407023.3409314

[31] Mubashir Husain Rehmani, Fayaz Akhtar, Alan Davy, and Brendan Jennings. 2018. Achieving Resilience in SDN-Based Smart Grid: A Multi-Armed Bandit Approach. In *IEEE NetSoft 2018*. https://doi.org/10.1109/NETSOFT.2018.8459942

[32] Amedeo Sapio, Ibrahim Abdelaziz, Abdulla Aldilaijan, Marco Canini, and Panos Kalnis. 2017. In-Network Computation is a Dumb Idea Whose Time Has Come. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks* (Palo Alto, CA, USA) *(HotNets'17)*. https://doi.org/10.1145/3152434.3152461

[33] Ilias Siniosoglou, Panagiotis Radoglou-Grammatikis, Georgios Efstathopoulos, Panagiotis Fouliras, and Panagiotis Sarigiannidis. 2021. A Unified Deep Learning Anomaly Detection and Classification Approach for Smart Grid Environments. *IEEE Trans. Netw. Serv. Manag.* 18 (2021). Issue 2. https://shorturl.at/lpHSW

[34] Giuseppe Siracusano and Roberto Bifulco. 2018. In-network Neural Networks. In *The Conference on Systems and Machine Learning (SysML)*.

[35] Giuseppe Siracusano, Salvator Galea, Davide Sanvito, Mohammad Malekzadeh, Hamed Haddadi, Gianni Antichi, and Roberto Bifulco. 2022. Re-architecting Traffic Analysis with Neural Network Interface Cards. In *The 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI '22)*.

[36] Seyedeh Mahsan Taghavinejad, Mehran Taghavinejad, Lida Shahmiri, Mohammad Zavvar, and Mohammad Hossein Zavvar. 2020. Intrusion Detection in IoT-Based Smart Grid Using Hybrid Decision Tree. In *ICWR*. https://doi.org/10.1109/ICWR49608.2020.9122320

[37] Aaron Turner and Fred Klassen. 2013. Tcpreplay. https://tcpreplay.appneta.com/

[38] Paola Vargas and Iris Tien. 2023. Impacts of 5G on cyber-physical risks for interdependent connected smart critical infrastructure systems. *International Journal of Critical Infrastructure Protection* 42 (2023). https://doi.org/10.1016/J.IJCIP.2023.100617

[39] Wireshark. 1998. https://www.wireshark.org/docs/man-pages/tshark.html

[40] Guorui Xie, Qing Li, Yutao Dong, Guanglin Duan, Yong Jiang, and Jingpu Duan. 2022. Mousika: Enable General In-Network Intelligence in Programmable Switches by Knowledge Distillation. In *IEEE INFOCOM 2022*. https://doi.org/10.1109/INFOCOM48880.2022.9796936

[41] Wei Xiong, Yu Lei, Yuqing Zhou, Yachao Zhang, Su Wei, and Zhe Liu. 2022. A Smart Grid Traffic Anomaly Detector Based on Deep Learning. In *2022 International Conference on Frontiers of Communications, Information System and Data Science (CISDS)*. https://doi.org/10.1109/CISDS57597.2022.00015

[42] Zhaoqi Xiong and Noa Zilberman. 2019. Do Switches Dream of Machine Learning? Toward In-Network Classification. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks (HotNets '19)*. https://doi.org/10.1145/3365609.3365864

[43] Hanyu Zeng, Zhen Wei Ng, Pengfei Zhou, Xin Lou, David K.Y. Yau, and Marianne Winslett. 2022. Detecting Cyber Attacks in Smart Grids with Massive Unlabeled Sensing Data. In *SmartGridComm 2022*. https://shorturl.at/dmtzM

[44] Xiaoquan Zhang, Lin Cui, Fung Po Tso, and Weijia Jia. 2021. pHeavy: Predicting Heavy Flows in the Programmable Data Plane. *IEEE Trans. Netw. Serv. Manag.* 18, 4 (2021), 4353–4364. https://doi.org/10.1109/TNSM.2021.3094514

[45] Changgang Zheng, Xinpeng Hong, Damu Ding, Shay Vargaftik, Yaniv Ben-Itzhak, and Noa Zilberman. 2023. In-Network Machine Learning Using Programmable Network Devices: A Survey. *IEEE Commun. Surv. Tutor.* (2023). https://doi.org/10.1109/COMST.2023.3344351

[46] Changgang Zheng and Noa Zilberman. 2021. Planter: Seeding Trees within Switches. In *SIGCOMM '21* (Princeton, NJ, USA). https://shorturl.at/ozU46

[47] Guangmeng Zhou, Zhuotao Liu, Chuanpu Fu, Qi Li, and Ke Xu. 2023. An Efficient Design of Intelligent Network Data Plane. In *USENIX Security 23*. https://www.usenix.org/conference/usenixsecurity23/presentation/zhou-guangmeng