Toward a Reliable Network-Based Intrusion Detection Model for SCADA: A Classification with Reject Option Approach

Paulo Roberto de Oliveira*, Eduardo K. Viegas*, Altair O. Santin*, Pedro Horchulhack*, Everton de Matos†
*Pontifical Catholic University of Paraná, Curitiba, Paraná, 80215-901, Brazil.
{paulo.oliveira, eduardo.viegas, santin, pedro.horchulhack}@ppgia.pucpr.br
†Technology Innovation Institute, Abu Dhabi - United Arab Emirates.
everton.dematos@tii.ae

***Abstract*—Industrial control systems (ICS) are often targeted by highly motivated attackers seeking to disrupt their services due to its critical nature. Traditional cybersecurity does not provide the necessary reliability for ICS systems. Even when implemented with many layers of defense, including network intrusion detection systems (NIDS). This paper proposes a dynamic and reliable intrusion detection model that is implemented in two steps. First, it proactively classifies each type of possible network attack on the basis of the current network traffic behavior. Second, it evaluates the classification quality through rejection option, which is an indication of its reliability. By adapting to the evolving network traffic, our proposal increases the system robustness against motivated attackers. The proposed model effectiveness has been demonstrated by experimenting in a controlled testbed with more than 14 attack categories. The dynamic selection of security mechanisms allowed us to increase the detection accuracy by up to 26%. Moreover, the classification evaluation in the proposed model achieves up to 99% detection accuracy with only 1% rejection.**

***Index Terms*—ICS, SCADA, Intrusion Detection, Machine Learning**

I. Introduction

Industrial Control Systems (ICSs) are integrated architectures used to manage and control industrial processes and associated infrastructure in sectors such as manufacturing, energy, and transportation [1]. It integrates hardware, software, and network components to enable their automation and optimize industrial processes, including machinery, sensors, data acquisition, and communication interfaces. The Supervisory Control and Data Acquisition (SCADA) system is widely used in ICSs, as it enables the control and automation of Programmable Logic Controllers (PLCs) through an Human-Machine Interface (HMI), leveraging various protocols such as Modbus, DNP3, and OPC for seamless communication and data exchange. [2]. Securing ICSs systems such as SCADA is a must due to its critical role in infrastructure systems. For instance, their breach has been used in a power grid resulting in widespread power outages, disrupting daily life, and potentially causing economic and social losses [3].

Given their critical nature, attackers are highly motivated to disrupt ICSs systems, even utilizing multiple zero-day vulnerabilities to reach their goal [4]. In practice, attackers analyze their target ICS infrastructure over extended periods to effectively craft their attacks. As a result, a comprehensive array of security solutions, encompassing authentication, authorization, firewalls, and VPNs, must be implemented to ensure the comprehensive safeguarding of these systems. Network-based Intrusion Detection Systems (NIDSs) tools are commonly used by operators to evaluate ICS network traffic. To achieve intrusion detection, proposed schemes often employ *rule-based* or *behavior-based* approaches [5]. On the one hand, *rule-based* techniques detect intrusions based on a database of well-known attack patterns. Thus, although they can effectively detect well-known intrusions, they cannot protect systems against novel threats. On the other hand, *behavior-based* approaches perform intrusion detection by analyzing the event behavior and identifying deviations from a pre-established normal baseline. Therefore, it has garnered research interest as they promise to detect new attacks.

Over the last years, several highly accurate *behavior-based* NIDSs have been proposed in the literature, wherein authors typically implement their solution as a pattern recognition task making use of Machine Learning (ML) techniques [6]. To accomplish this objective, designed schemes develop a behavioral ML model by evaluating the behavior available in a training dataset. The accuracy of the obtained model is then assessed through a testing dataset, which is expected to be experienced when the system is deployed in production settings [7]. Unfortunately, despite the promising results reported in the literature, ML-based intrusion detection schemes designed for safeguarding ICS systems are seldom implemented in practice [8]. Conversely, it remains primarily a research topic with limited practical implementation.

ICS systems, including SCADA, are the target of highly motivated attackers who meticulously study their targets to launch successful attacks [9]. Consequently, stationary ML-based intrusion detection systems can be easily evaded by attackers, as the parameters of the deployed model can only be modified after undergoing a new training process. However, even if the model's behavior is not easily assessable by attackers, as time passes, it will inevitably lead to the emergence of new attacks, thereby increasing the likelihood

of attackers evading the deployed ML model. This is because current ML-based techniques do not assess the quality of their classifications, leaving the network operator unaware of unreliable outputs [10]. Ideally, used ML-based NIDSs must be able to proactively select the security mechanisms that should be used to improve the system's security.

Contribution. In light of this, this paper introduces a new intrusion detection scheme for ICS, aiming to enhance classification quality by dynamically adjusting the deployed model configuration based on the current system's input. The proposal is split into two phases. First, classification is executed using a dynamic classifier selection scheme that intelligently selects the best suitable classifiers based on the current event behavior. Our insight is to enhance the classification quality by dynamically adjusting the system's classification configuration based on the evaluated event features. Second, the classification quality is evaluated using a classification with a reject option approach. As a result, our system can identify potential misclassifications made by the model, enhancing its reliability.

The main contributions of this paper are:

- A new publicly available SCADA dataset containing realistic network traffic. The dataset contains 14 network attack categories, assessing 3 different ICS security solutions;
- A new dynamic and reliable intrusion detection model that proactively selects the appropriate security solution and effectively detects potential misclassifications. The proposed scheme reaches up to 99% average accuracy with only 1% rejection rate.

Roadmap. The remainder of this paper is organized as follows. Section II introduces the fundamentals behind ICSs and intrusion detection schemes. Section III describes the current state of the art on ICS intrusion detection. Section IV outlines our proposed scheme, while Section VI presents its evaluation. Finally, Section VII concludes our work.

II. Background

This section provides an in-depth description of ICS, encompassing SCADA and its main associated infrastructure components. Furthermore, it introduces the key elements of ML typically used in NIDSs.

A. Industrial Control Systems

An ICS is a complex, usually legacy, system used to control and manage a collection of hardware, software, and network elements related to an industrial complex [1]. It is commonly utilized by critical infrastructure companies operating in various fields, including manufacturing, oil and gas, energy, water treatment, and more. To accomplish this objective, a typical ICS is composed of three main components, namely SCADA, PLC, and communication components. SCADA aims at managing and controlling the industrial complex assets, including its sensors, equipment, and devices, through an HMI. It facilitates data collection and storage of assets, generating and collecting alerts, and the capability to issue control commands. PLCs are digital devices responsible for controlling industrial machinery or processes. They serve as an interface between the SCADA system or other associated controllers, facilitating interaction with sensors and actuators. Finally, the communication components utilize various communication protocols, such as Ethernet, Profibus, and Modbus, to facilitate the interaction between the SCADA system and the PLC.

Hence, ICSs are inherently cyber-connected systems, rendering them vulnerable to highly motivated attackers who seek to disrupt the functioning of critical industrial infrastructure [11]. A vulnerable ICS infrastructure allows attackers to gain control over industrial PLCs, posing risks to industry assets, disrupting their services, and resulting in financial and social losses. For example, in 2015 BlackEnergy malware infected Ukrainian SCADA systems used in a power grid ICS, resulting in widespread power outages affecting over 230 thousand consumers [12]. Unfortunately, securing ICSs is not an easily achievable task, as their architecture typically comprises interconnected components that can not be easily updated [13]. Consequently, ICS operators must employ multiple security components, including authentication, authorization, firewalls, VPNs, and NIDS, to fortify the underlying architecture.

B. Network-based Intrusion Detection System

Behavior-based NIDSs are typically used as a security mechanism to monitor and analyze network activities. Solutions are generally executed as a pattern recognition task implemented through ML techniques [14]. The entire process comprises four main modules. Firstly, the *Data Acquisition* module continuously gathers network events from the monitored environment, such as collecting network packets from a Network Interface Card (NIC). Secondly, the *Feature Extraction* module extracts the behavior of the collected data and compiles a feature vector. The behavior of network events is generally represented as a network flow, summarizing the communication between two network entities within a given time window. For instance, computing the number of exchanged network packets between two hosts over the last 15 seconds. The extracted feature set serves as input for the *Classification* module, which utilizes a ML model to classify the input as either *normal* or *intrusion*. Lastly, the *Alert* module signals events classified as intrusions to the network operator.

In practice, the reliability of designed behavior-based NIDSs relies on the quality of the underlying used ML model [15]. Unfortunately, creating a realistic intrusion dataset that accurately represents the characteristics of ICS systems is not easily achievable. The behavior of ICS systems varies widely depending on the associated industrial assets, requiring ML-based NIDSs can accurately capture this variability [16]. Notwithstanding, due to the critical nature of these systems, attackers are highly motivated to evade the reliability of designed security mechanisms, often employing a wide range of zero-day attacks to compromise their security. Yet, existing ML-based NIDSs for ICSs often overlook these challenges and rely on traditional pattern recognition approaches, which are

known to be unable to achieve the necessary level of reliability required for industrial architectures [17].

III. Related Works

Over the last years, several works have proposed highly accurate ML-based techniques for NIDSs. Despite the promising reported results, proposed schemes are seldom implemented in production environments, where network operators typically opt for traditional misuse-based approaches. The network behavior experienced in production ICSs is highly dynamic due to the nature of industrial assets, making the design of a realistic dataset a challenging task [18]. Notwithstanding, the designed techniques must be capable of adapting to changes in attacker behavior, driven by their strong motivation to disrupt the critical nature of monitored industrial assets.

In general, the approaches proposed in the literature prioritize efforts to enhance the system's accuracy. For instance, Ahakonye *et al.* [19] proposes a feature selection technique for ML-based NIDS in SCADA systems. The authors improve their false-positive rates when proactively selecting their model features in outdated NIDS datasets. Unfortunately, the applicability of their proposed model in real-world ICSs is overlooked. Similarly, Y. Ouyang *et al.* [20] proposes NIDS implemented using a few-shot learning scheme for SCADA systems. Their proposed scheme improved detection accuracy compared to other approaches on a SCADA-related dataset. The impact of motivated attackers on circumventing their proposed model is not evaluated. R. Grammatikis *et al.* [21] addressed the lack of reliability in current ML-based NIDSs in the literature by developing a protocol-specific intrusion detection scheme. Their model was able to significantly improve the system's accuracy when compared to traditional techniques. However, the protocol-specific nature of the scheme may lead to a lack of generalization, thus, unreliability to be used in production environments. Similarly, S. Alem *et al.* [22] proposed a more specific intrusion detection scheme for ICSs aiming PLC intrusion detection. The authors use a neural network that evaluates PLC-related messages. Although the authors assess the quality of their proposed scheme in a real industrial environment, the applicability to other industrial assets is not evaluated.

The lack of reliability in ICS intrusion detection is rarely considered in related works. E. Anthi *et al.* [23] proposed a three-layered ICS intrusion detection scheme aiming for higher detection effectiveness. Their model significantly improved accuracy when the detection layers were jointly used. Unfortunately, their evaluation uses an unrealistic intrusion dataset that does not realistically depict the ICS nature. J. Song *et al.* [24] proposed an intrusion detection scheme for energy ICSs that makes use of temporal pattern classification coped with anomaly detection. The authors claim to improve the reliability of their system by assessing the temporal nature of ICS events. Despite the scheme's improvement in accuracy compared to other techniques, the study did not assess the model's quality in the presence of highly motivated attackers. L. Rosa *et al.* [25] proposed a new feature aggregation scheme

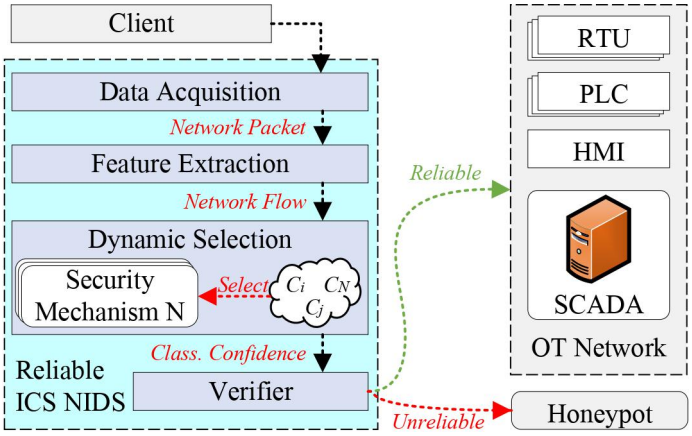


Fig. 1: A reliable intrusion detection model for industrial control systems. The proposed scheme proactively selects the most suitable security mechanisms tailored for the current network traffic.

tailored for ICSs. Their approach improves the detection performance when compared to related work. However, it overlooks the detection reliability in the presence of new kinds of attacks. J. Lopes et al. [26] proposed intrusion detection targeted explicitly at the IEC 61850 protocol. Their approach improves detection reliability by assessing the specificities of the IEC 61850 protocol. The detection on other kinds of network protocols are not addressed. Similarly, S. Mubarak *et al.* [27] concentrate on detecting the Modbus protocol, enhancing detection accuracy while tailoring their detection module to evaluate a single protocol.

As a result, current approaches in the literature often fail to address the detection reliability required by current ICSs infrastructures. This is because, in production environments ICS, attackers are motivated to circumvent the reliability of used security mechanisms. As a consequence, the employed detection approaches must be capable of handling a broader range of challenges during the detection phase, including the detection of new attacks and proactively selecting the security mechanisms to address incoming threats.

IV. A Reliable Intrusion Detection Model for Industrial Control Systems

To address the lack of reliability in current ML-based NIDS for ICSs, we propose a reliable intrusion detection model. The proposal's goal is to select the most appropriate security mechanisms that should be employed to detect the current network traffic behavior. In practice, the system can adapt to the current Operational Technology (OT) incoming network traffic, hence, increasing the system's robustness to highly motivated attackers. Notwithstanding, our proposed mechanism assesses the quality of the chosen security mechanisms to evaluate the current network traffic, signaling potentially unreliable classifications. Figure 1 shows our proposed model overview, comprising two main modules: *Dynamic Selection* and *Verifier*.

The *Dynamic Selection* module goal is to adjust the used security mechanisms based on the current OT network traffic. The model assesses the network traffic behavior and intelligently selects a subset of security mechanisms that are most suitable for classifying the incoming network traffic behavior. Our main insight is to select which security tools should be used in a proactive manner to ensure that our system can maintain its classification reliability. Consequently, the proposed model can enhance the system's accuracy, even when facing dynamic attacker behavior.

The *Verifier* module goal is to ensure the system's reliability is maintained as time passes, even in the presence of new network-based attacks. To achieve such a goal, the decision performed by the *Dynamic Selection* module is assessed through classification with a reject option. The primary objective is to accept only highly confident classifications, rejecting unreliable counterparts as time progresses. As a result, our proposed scheme can dynamically adapt based on the current OT network traffic while maintaining the system's reliability over time.

The following subsections provide a more detailed description of our proposed model architecture, including the modules that implement it.

A. Dynamic Selection

Due to their critical nature, ICS systems are frequently targeted by highly motivated attackers who often exploit multiple zero-day vulnerabilities to achieve their objectives. As a result, deployed architectures are designed to incorporate multiple security mechanisms to improve the system's security, such as firewalls, *rule-based* and *behavior-based* NIDSs. Despite such efforts, the static behavior of deployed security mechanisms makes them vulnerable to circumvention by attackers. This is because, over an extended period, ranging from days to weeks, attackers can easily deduce the security rules of the ICS, exploiting them as necessary to launch their attacks effectively. In light of this, the *Dynamic Selection* module's goal is to proactively assess the ICS network traffic behavior to select the most suitable security mechanisms to be used. As a result, the system can effectively address the detection of new types of attacks by assessing and selecting the most appropriate security mechanisms to detect them.

To implement the *Dynamic Selection* module, we consider the selection of the security mechanisms as a dynamic selection of classifier task. In practice, we build a dynamic selection of classifier model based on the events correctly classified by each security mechanism. Algorithm 1 shows the overview of our model-building procedure. It receives as input a set of security mechanisms (SM), a previously labeled network event dataset ($\mathcal{D}$), and a dynamic selection model (h). The algorithm iterates through the dataset $\mathcal{D}$ to build the dynamic selection label ($\mathcal{Y}$), which maps the security mechanisms that correctly classify the input event. The dynamic selection model (h) is built based on the obtained label $\mathcal{Y}$. As a result, the model is fine-tuned to identify the security mechanisms that can accurately classify the evaluated network event.

Algorithm 1 Dynamic Security Mechanism Building

Require:
Security Mechanisms SM
Network Event Dataset $\mathcal{D}$
Dynamic Selection Model h
procedure MODEL BUILDING(SM, D, h)
  $\mathcal{Y} \leftarrow \emptyset$ ▷ Dynamic Selection Label
  for $x_i \in \mathcal{D}$ **do**
    $y_{sm} \leftarrow \emptyset$
    for $sm_i \in SM$ **do** ▷ Find Correct Mechanisms
      $y \leftarrow apply(sm_i, x_i)$
      if $label(x_i)$ is y **then**
        $y_{sm} \leftarrow y_{sm} + sm_i$
      end if
    end for
    $\mathcal{Y} \leftarrow \mathcal{Y} + \{x_i, y_{sm}\}$ ▷ Label Correct Mechanisms
  end for
  return $h.fit(\mathcal{D}, \mathcal{Y})$ ▷ Fit Dynamic Selector
end procedure

During production deployment, the resulting model is employed as a security mechanism selector (Fig. 1, *Dynamic Selection*). In practical terms, the model output is a list of security mechanisms that should be employed to detect the current network event. Consequently, the *Dynamic Selection* module can adjust the security mechanisms based on the current network traffic behavior. This approach enhances our system's reliability as the security mechanisms are proactively selected based on the evaluated network behavior. The selected security mechanisms coped with the classification confidence value are used as input to the subsequential *Verifier* module (Fig. 1).

B. Verifier

Given their exposure to motivated attackers, ICSs are often the target of zero-day attacks. As a result, despite selecting the most suitable set of security mechanisms, as time passes, attackers will inevitably find ways to circumvent the reliability of the current security solutions in use. To tackle this challenge, the goal of the *Verifier* module is to assess the quality of the current security solutions in detecting the network traffic accurately. In practice, the module evaluates the quality of the decisions made by the *Dynamic Selection* module, alerting the network operator about unreliable security mechanism selections. Based on the assessed decisions, the operator may take counteractions, such as redirecting the identified network traffic to a honeypot or utilizing it to update the rules of the deployed security mechanisms. To achieve this goal, the *Verifier* module introduces a classification with reject option to the outputs of the *Dynamic Selection* module.

Classification with a reject option implements the rejection decision function on a given input event by combining the used model h with a rejector r. This approach allows the system to reject classifying certain events when their confidence levels do not meet a predefined threshold, reducing the likelihood of

misclassifications. The model h produces classification confidence values $\alpha = \{\alpha_{normal}, \alpha_{sm}\}$ for each input event x. The α_{normal} denotes the event confidence to belong to the *normal* class, and α_{sm} the event confidence to belong to *attack*, hence, should be forwarded to a set of security mechanisms, such that $\alpha \in \mathbb{R}[0, 1]$. The rejector r accepts or rejects the classification based on its associated confidence values α. Finally, the model h implementation is coped with the rejection function, as determined by the following equation:

$$h(x,t) \begin{cases} \emptyset & \text{if } \alpha \leq t \\ h(x) & \text{otherwise} \end{cases} \tag{1}$$

where $\emptyset$ denotes events likely to be incorrect decisions the predictor performs. Therefore, our proposal assesses the model's classification confidence values α to implement the rejector in a classifier agnostic rationale to achieve such a goal. Equation 2 shows the rejector implementation function in our proposal.

$$d(h,x,t) \begin{cases} \overbrace{h(x,t_{normal})}^{\text{Normal Classified}} & \text{if } \alpha_{normal} > \alpha_{sm} \\ \underbrace{h(x,t_{sm})}_{\text{SM Classified}} & \text{otherwise} \end{cases} \tag{2}$$

where t denotes the pair of acceptance threshold values for each class such that $t \in \mathbb{R}[0, 1]$, and h a decision function coped with the proposal *Verifier* module.

Consequently, the *Verifier* module establishes the input events acceptance based on the *Dynamic Selection* classification confidence level α, which is assessed based on the acceptance threshold t. The threshold must be defined based on the network operator's discretion. A low acceptance threshold will result in fewer network events being rejected, thereby increasing the system's exposure to potential threats. Conversely, a higher acceptance threshold will lead to the rejection of more network events, enhancing the system's reliability but requiring more intervention from the network operator. Our proposal finds the optimal acceptance threshold by solving the following equation:

$$T(h, \mathcal{D}, y) = \underset{t^{\times 2} \in \mathbb{R}[0,1]}{\arg\min}\, Error(h, \mathcal{D}, y, t) + Rej(h, \mathcal{D}, y, t) \tag{3}$$

where $T(h, \mathcal{D}, y)$ is a threshold finding function for model h on dataset $\mathcal{D}$, with a label y, while the functions $Error$, and Rej measures the respectively error and rejection rates of the model h, on dataset $\mathcal{D}$, with a label y while using threshold t. As a result, we determine the optimal rejection threshold by balancing the minimum error and rejection rate.

C. Discussion

Our proposed model aims to enhance the reliability of ICS NIDS by proactively selecting the deployed security mechanisms (Section IV-A) and simultaneously identifying unreliable decisions (Section IV-B). On the one hand, the

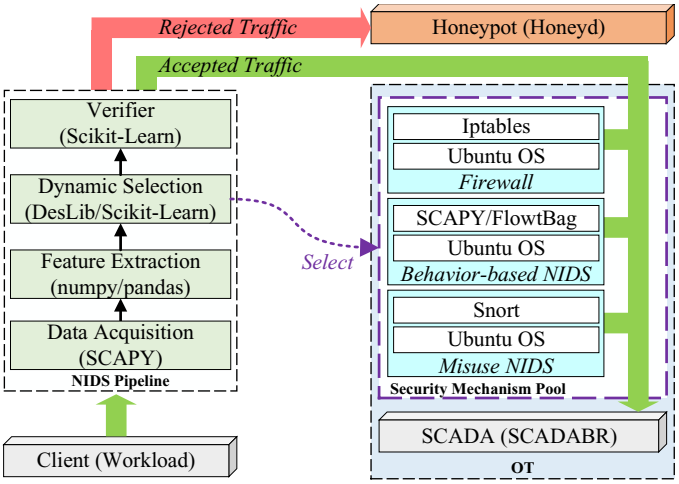


Fig. 2: Implemented proposal prototype and used testbed throughout the conducted experiments.

Dynamic Selection module enhances the system's robustness against motivated attackers who exploit variable attacker behavior to undermine the reliability of deployed security mechanisms. The module dynamically selects the security mechanisms used to detect the current network traffic behavior, thereby increasing the challenge for attackers to evade the designed security tools. On the other hand, the *Verifier* module evaluates the quality of decisions made by the preceding module, effectively signaling unreliable decisions to the network operator. The main insight of the proposal is to enhance the system's robustness against highly motivated attackers seeking to disrupt the security of ICS systems.

V. Model Implementation

This section elaborates further on our model implementation, including the APIs utilized and the considered architecture. Furthermore, it describes the deployed testbed used to evaluate our proposed method.

A. Prototype

We implemented and deployed a proposal prototype in a distributed environment, as illustrated in Figure 2. We considered a SCADA system deployed through ScadaBR $v.1.0CE$. In addition, we deployed a honeypot through the honeyd $v.1.5c$, which mirrors the incoming network traffic to another ScadaBR server. The honeypot is used in our proposal to receive rejected network traffic by our *Verifier* module. The incoming OT network traffic can be evaluated by three security tools, as follows:

- ***Firewall***. A set of iptables rules deployed on top of an Ubuntu $v.22.04$ OS. The firewall rules were defined based on related works [28]
- ***Behavior-based NIDS***. A ML-based NIDS implemented using a Naive Bayes classifier. The classifier was implemented using the likelihood of the features as Gaussian on top of scikit-learn API $v.0.24$. The network flow features were extracted using FlowTBag feature extractor [29].

TABLE I: Generated network traffic throughout our testbed.

#	Behavior (Tool)	Net. Pkts.	Exec. Time (m)
1	XSS (Acunetix)	6k	1028
2	Code Injection (Arachni)	17k	392
3	Read Register (Smod)	58k	153
4	DOS Write Register (Smod)	2k	133
5	DOS Write Coils (Smod)	1k	4
6	Portscan (Nmap)	140k	33
7	Write Single Coils (Smod)	45k	52
8	Advanced scan (Nessus)	35k	33
9	Read Input Register (Smod)	150k	56
10	Scanner UID (Smod)	38k	86
11	SQL Injection (Arachni)	4k	70
12	Read Coils (Smod)	320k	20
13	Scanner (Smod)	2k	113
14	Write Register (Smod)	296k	164
-	Normal (Workload)	5.1B	5760

The procedure used for the ML model training was conducted similarly to our proposal, further described in Section VI-A.

- ***Misuse-based NIDS***. A signature-based NIDS implemented through Snort tool $v.3.0$, using the $snort3$ community rules.

Our proposal (Fig. 2, NIDS pipeline) was implemented to assess the incoming SCADA network traffic. To simulate various OT-related network traffic, we utilized several well-known workload tools, generating traffic for protocols such as Modbus, DNP3, OPC, and web service requests to SCADA. The generated network traffic is collected by our *Data Acquisition* module implemented using SCAPY $v.2.5.0$. Similarly, we extract the network features using FlowTBag feature extractor [29], which summarizes the network communication in 15 second time-window intervals. We implement the *Dynamic Selection* module through the DESLib API $v.0.4$, and assess the classification confidence on the *Verifier* module through the *predict proba* function. The model building details are outlined in Section VI-A.

B. Testbed

The described prototype was deployed in a controlled testbed designed for data collection purposes. To accomplish such a task, we deploy 100 client workload machines to generate the normal network traffic. At randomly varying intervals, we generate attacker-related network traffic aiming the deployed SCADA. At total we use 14 attacker machines that generate specific network attacks. The testbed was executed for a total of 96 hours, the generated network traffic is shown on table I.

Throughout the testbed execution, we logged the outputs of the deployed security mechanisms (Fig 2, *Firewall*, *Behavior-based NIDS*, and *Misuse-based NIDS*). The outputs are used to build our *Dynamic Selection* module, further described in Section VI-A. In addition, the generated network traffic is stored in a PCAP format to enable the later assessment of our solution.

VI. Evaluation

This section evaluates the proposed model for reliable network-based intrusion detection in ICS by addressing four main research questions:

- (***RQ1***) *How does traditional security mechanisms detect ICS attacks?*
- (***RQ2***) *Can our proposal dynamic selection improve the system accuracy?*
- (***RQ3***) *Can our proposed classification assessment technique identify unreliable classifications?*

The following subsections outline the construction of the used models in the evaluation and describe their performance in the testbed.

A. Model Building

The performance of the proposed model, as depicted in Figure 1, was evaluated using the testbed described previously (see Section V-B). To achieve such a goal, we consider a scenario in which new attacks are generated over time by building three datasets as follows:

- *Training*. Attacks from 1^{st} to 5^{th} and 40% of normal traffic are utilized for training (Table I).
- *Validating*. Attacks from 6^{th} to 10^{th} and 30% of normal traffic are utilized for testing (Table I).
- *Testing*. Attacks from 11^{th} to 14^{th} and 30% of normal traffic are utilized for training (Table I).

The *training* dataset is used for model training, the *validation* dataset for model fine-tuning, and *testing* dataset for measuring the model accuracies. We evaluate our proposed *Dynamic Selection* implemented with three distinct dynamic selection classifiers, namely k-Nearest Oracle-Eliminate (KNORA-E), k-Nearest Oracle Union (KNORA-U), and k-Nearest Output Profile (KNOP). The classifiers are implemented using a *bagging* pool of 100 estimators with replacement event selection. We also use 11 neighbors to estimate the competence region and k-Nearest Neighbor (KNN) for distance computation. The parameters were empirically set, and no significant influence on the results was observed when varying them. The classifiers were implemented on DESLib API $v.0.4$. Similarly, we train the *Behavior-based NIDS* (Fig. 2) using the Gaussian Naive Bayes implemented on top of *scikit-learn*.

Both our proposed *Dynamic Selection* and the *Behavior-based NIDS* were trained on the *training* dataset. The datasets were built by extracting network flow-based features using the FlowTBag feature extractor [29], which summarizes the network communication in 15 second time-window intervals. Due to the highly unbalanced nature of the training datasets, with the majority of events being classified as *normal*, we apply a random undersampling without replacement technique. The resulting dataset is normalized using a minimum versus maximum range normalization approach, which scales the values to a range from 0 to 1.

TABLE II: Detection accuracy for each network traffic on our testbed according to the detection approach.

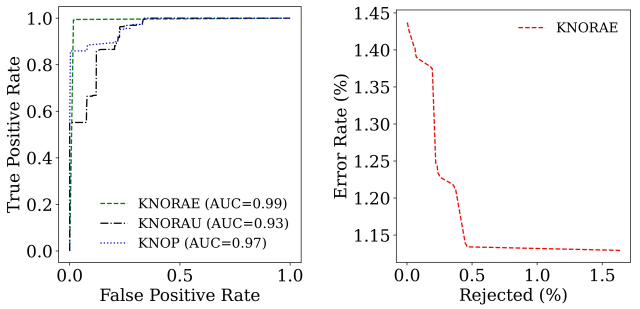
Behavior	Detection Accuracy (%)					
	Traditional			*Dynamic Selection*		
	Firewall	**Behavior NIDS**	**Misuse NIDS**	**KNORA-U**	**KNORA-E**	**KNOP**
XSS	40.5	50	90.9	48.0	100	59.6
Code Injection	50	95.6	95.6	99.0	100	98.8
Read Reg.	96.6	99.7	50	100	98.5	100
DOS Write Reg.	72.5	97.2	50	100	95.5	100
DOS Write Coils	88.2	50	50	100	50.8	100
Portscan	50	50	100	100	100	100
Write Single Coils	88.2	50	50	100	100	100
Advanced scan	44.9	50	98	99.9	99.6	99.9
Read Input Reg.	96.6	99.7	50	100	100	100
Scanner UID	44.3	50	50	100	100	100
SQL Injection	50	50	81.9	50.9	100	50.9
Read Coils	95.3	99.5	50	100	99.7	100
Scanner	52.6	50	100	50	50	50
Write Reg.	91.6	99.7	50	100	100	100
Normal	100	100	100	87.4	62.4	85.9

B. ICS Intrusion Detection

To answer *RQ1* we first investigate the detection performance of the selected security mechanisms (Fig. 2, *Security Mechanism Pool*). To achieve such a goal, we measure the intrusion detection performance on our previously executed testbed (Table I). The evaluation goal is to assess the reliability of commonly used ICS security mechanisms. Table II shows the detection accuracies for each generated network attack and used security mechanism (*Traditional*). In practice, we observed that all the assessed traditional approaches had their detection reliability impacted by at least a subset of the examined attacks. As an example, when aiming for a detection accuracy of 95%, the evaluated security solutions achieved their objective for only 3, 6, and 4 out of the 14 of the examined attacks for the *Firewall*, *Behavior-based NIDS*, and *Misuse-based NIDS*, respectively. The results suggest that the evaluated security solutions lacked the required detection reliability for ICSs. In addition, the variation in accuracy for different evaluated attacks, observed through the distinct detection performance of each security mechanism for various attack categories, highlights the potential for our solution to enhance detection accuracy when security mechanisms are appropriately selected based on the network traffic behavior.

To answer *RQ2*, we further investigate how our *Dynamic Selection* approach can further improve the system's detection reliability. To accomplish this objective, the selected dynamic classifiers (see Section VI-A) are trained to select the security mechanisms that correctly classify the network event (Alg. 1). During the implementation, we adjust the training label based on the correctly classified events by each security mechanism and subsequently evaluate the obtained model on the complete testing dataset.

Table II shows the detection accuracy for our *Dynamic Selection* module, as measured by the accuracy of selecting



(a) ROC Curve (b) KNORA-E Error $vs.$ Reject

Fig. 3: Detection performance of our proposed dynamic selection model with and without the classification assessment.

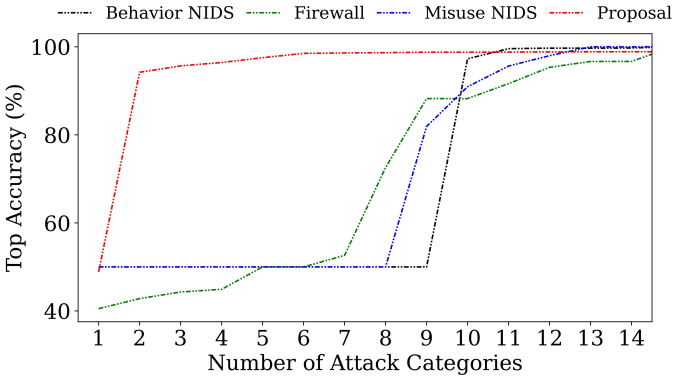


Fig. 4: Sorted top detection accuracy for each selected intrusion detection technique.

the security mechanism that correctly classified the network event. The results indicate that our proposed approach, which proactively selects the security mechanism, can significantly increase the system's detection accuracy. For example, when aiming for a detection accuracy of 95%, the most accurate dynamic classifier (KNORA-E) could adequately identify 11 out of the 14 attacks, reaching an average accuracy of 90%, while the traditional techniques reach an average accuracy of only up to 71% (*Behavior-based NIDS)*, an increase of 26.7%. In practice, our proposed model effectively identified which security mechanism should be used to detect the evaluated network traffic.

Finally, we answer *RQ3* by investigating how the proposal *Dynamic Selection* coped with the *Verifier* module can increase the system accuracy further (see Fig. 1). To accomplish this task, we evaluate the KNORA-E, which has been identified as the most accurate dynamic selection classifier, as shown in Fig. 3a. To find the optimal acceptance threshold for both *normal* and *sm* classes, we use the Class Related Threshold (CRT) in conjunction with the KNORA-E as the baseline. The thresholds are varied in increments of 0.01, ranging from 0.0 to 1.0, to solve Eq. 3. Figure 3b shows the error $vs.$ rejection threshold for the KNORA-E classifier on the *testing* dataset. Our proposed *Verifier* module significantly increases system accuracy with minimal trade-off in terms of rejections. For instance, the *Verifier* module achieves a 1% error rate with a

rejection rate of only half a percent.

We compare our solution with the traditional security mechanisms to assess the proposal's performance. Figure 4 shows the sorted detection accuracy of each evaluated detection technique for the testbed attacks. Our proposed model with the *Verifier* technique reaches a 95% detection accuracy for 12 out of the 14 evaluated attacks, while the traditional approaches are only able to reach the same level of accuracy for 5 attack categories (Table II, *Behavior-based NIDS*).

VII. Conclusion

Given their critical nature, attackers are highly motivated to disrupt ICSs systems, demanding that used security mechanisms can reach high detection accuracies. Yet, current security mechanisms are unable to reach the needed level of reliability for ICS deployment. In light of this, this paper proposed a new dynamic selection model for NIDSs aiming for improved detection accuracies while ensuring system reliability in the presence of new attack categories. Our proposal enables the proactive selection of security mechanisms based on the current network traffic behavior. In addition, we can further enhance the system reliability by evaluating the detection quality based on the classification confidence values. The experiments have demonstrated that our proposal surpasses traditional security solutions and provides the required level of detection reliability for ICS deployment.

Acknowledgment

This work was partially sponsored by Brazilian National Council for Scientific and Technological Development (CNPq) grants n. 304990/2021-3 and n. 407879/2023-4.

References

[1] H. Kayan, M. Nunes, O. Rana, P. Burnap, and C. Perera, "Cybersecurity of industrial cyber-physical systems: A review," *ACM Comput. Surv.*, vol. 54, no. 11s, sep 2022.

[2] M. Alanazi, A. Mahmood, and M. J. M. Chowdhury, "Scada vulnerabilities and attacks: A review of the state-of-the-art and open issues," *Computers & Security*, vol. 125, p. 103028, 2023.

[3] W. Alsabbagh, S. Amogbonjaye, D. Urrego, and P. Langendörfer, "A stealthy false command injection attack on modbus based scada systems," in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, 2023, pp. 1–9.

[4] C. Sheng, Y. Yao, W. Li, W. Yang, and Y. Liu, "Unknown attack traffic classification in scada network using heuristic clustering technique," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023.

[5] R. R. dos Santos, E. K. Viegas, A. O. Santin, and P. Tedeschi, "Federated learning for reliable model updates in network-based intrusion detection," *Computers amp; Security*, vol. 133, p. 103413, Oct. 2023.

[6] P. Horchulhack, E. K. Viegas, A. O. Santin, F. V. Ramos, and P. Tedeschi, "Detection of quality of service degradation on multi-tenant containerized services," *Journal of Network and Computer Applications*, vol. 224, p. 103839, Apr. 2024.

[7] E. K. Viegas, A. O. Santin, and P. Tedeschi, "Toward a reliable evaluation of machine learning schemes for network-based intrusion detection," *IEEE Internet of Things Magazine*, no. 2, pp. 70–75, 2023.

[8] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Aug. 2022, pp. 3971–3988.

[9] J. Qian, X. Du, B. Chen, B. Qu, K. Zeng, and J. Liu, "Cyber-physical integrated intrusion detection scheme in scada system of process manufacturing industry," *IEEE Access*, vol. 8, pp. 147 471–147 481, 2020.

[10] J. Geremias, E. K. Viegas, A. O. Santin, A. Britto, and P. Horchulhack, "Towards a reliable hierarchical android malware detection through image-based cnn," in *2023 IEEE 20th Consumer Communications amp; Networking Conference (CCNC)*. IEEE, Jan. 2023.

[11] A. T. A. Ghazo and R. Kumar, "Critical attacks set identification in attack graphs for computer and scada/ics networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–0, 2023.

[12] M. Geiger, J. Bauer, M. Masuch, and J. Franke, "An analysis of black energy 3, crashoverride, and trisis, three malware approaches targeting operational technology systems," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1537–1543.

[13] T. Sasaki, A. Fujita, C. H. Ganán, M. van Eeten, K. Yoshioka, and T. Matsumoto, "Exposed infrastructures: Discovery, attacks and remediation of insecure ics remote management devices," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 2379–2396.

[14] P. Horchulhack, E. K. Viegas, and A. O. Santin, "Detection of service provider hardware over-commitment in container orchestration environments," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. IEEE, Dec. 2022.

[15] J. Geremias, E. K. Viegas, A. O. Santin, A. Britto, and P. Horchulhack, "Towards multi-view android malware detection through image-based deep learning," in *2022 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, May 2022.

[16] E. Viegas, A. Santin, J. Bachtold, D. Segalin, M. Stihler, A. Marcon, and C. Maziero, "Enhancing service maintainability by monitoring and auditing sla in cloud computing," *Cluster Computing*, vol. 24, no. 3, p. 1659–1674, Nov. 2020.

[17] R. R. dos Santos, E. K. Viegas, and A. O. Santin, "A reminiscent intrusion detection model based on deep autoencoders and transfer learning," in *IEEE Global Communications Conference*, 2021.

[18] S. V. B. Rakas, M. D. Stojanović, and J. D. Marković-Petrović, "A review of research work on network-based scada intrusion detection systems," *IEEE Access*, vol. 8, pp. 93 083–93 108, 2020.

[19] L. A. C. Ahakonye, C. I. Nwakanma, J.-M. Lee, and D.-S. Kim, "Agnostic ch-dt technique for scada network high-dimensional data-aware intrusion detection system," *IEEE Internet of Things Journal*, vol. 10, no. 12, pp. 10 344–10 356, 2023.

[20] Y. Ouyang, B. Li, Q. Kong, H. Song, and T. Li, "Fs-ids: A novel few-shot learning based intrusion detection system for scada networks," in *ICC 2021 - IEEE International Conference on Communications*, 2021.

[21] P. Radoglou-Grammatikis, P. Sarigiannidis, G. Efstathopoulos, P.-A. Karypidis, and A. Sarigiannidis, "Diderot: An intrusion detection and prevention system for dnp3-based scada systems," *International Conference on Availability, Reliability and Security (ARES)*, 2020.

[22] S. Alem, D. Espes, L. Nana, E. Martin, and F. De Lamotte, "A novel bi-anomaly-based intrusion detection system approach for industry 4.0," *Future Generation Computer Systems*, vol. 145, pp. 267–283, 2023.

[23] E. Anthi, L. Williams, P. Burnap, and K. Jones, "A three-tiered intrusion detection system for industrial control systems," *Journal of Cybersecurity*, vol. 7, no. 1, p. tyab006, 02 2021.

[24] J. Song, B. Li, Y. Wu, Y. Shi, and A. Li, "Real: A new resnet-alstm based intrusion detection system for the internet of energy," in *IEEE Conference on Local Computer Networks (LCN)*, 2020.

[25] L. Rosa, T. Cruz, M. B. de Freitas, P. Quitério, J. Henriques, F. Caldeira, E. Monteiro, and P. Simões, "Intrusion and anomaly detection for the next-generation of industrial automation and control systems," *Future Generation Computer Systems*, vol. 119, pp. 50–67, 2021.

[26] J. A. Lopez, I. Angulo, and S. Martinez, "Substation-aware. an intrusion detection system for the IEC 61850 protocol." in *Proceedings of the 17th International Conference on Availability, Reliability and Security*. ACM, Aug. 2022.

[27] S. Mubarak, M. H. Habaebi, M. R. Islam, F. D. A. Rahman, and M. Tahir, "Anomaly detection in ICS datasets with machine learning algorithms," *Computer Systems Science and Engineering*, vol. 37, no. 1, pp. 33–46, 2021.

[28] A. Sahu, P. Wlazlo, N. Gaudet, A. Goulart, E. Rogers, and K. Davis, "Generation of firewall configurations for a large scale synthetic power system," in *2022 IEEE Texas Power and Energy Conference (TPEC)*, 2022, pp. 1–6.

[29] L. C. B. Guimarães, G. A. F. Rebello, G. F. Camilo, L. A. C. de Souza, and O. C. M. B. Duarte, "A threat monitoring system for intelligent data analytics of network traffic," *Annals of Telecommunications*, vol. 77, no. 7-8, pp. 539–554, Oct. 2021.