A Host-based Intrusion Detection Model Based on OS Diversity for SCADA

Bruno B. Bulle, Altair O. Santin, Eduardo K. Viegas, Roger R. dos Santos
Graduate Program in Computer Science - Pontifical Catholic University of Parana, Brazil
{bruno.bulle, santin, eduardo.viegas, robson.roger}@ppgia.pucpr.br

***Abstract*—Supervisory Control and Data Acquisition (SCADA) systems have been a frequent target of cyberattacks in Industrial Control Systems (ICS). As such systems are a frequent target of highly motivated attackers, researchers often resort to intrusion detection through machine learning techniques to detect new kinds of threats. However, current research initiatives, in general, pursue higher detection accuracies, neglecting the detection of new kind of threats and their proposal detection scope. This paper proposes a novel, reliable host-based intrusion detection for SCADA systems through the Operating System (OS) diversity. Our proposal evaluates, at the OS level, the SCADA communication over time and, opportunistically, detects, and chooses the most appropriate OS to be used in intrusion detection for reliability purposes. Experiments, performed through a variety of SCADA OSs front-end, shows that OS diversity provides higher intrusion detection scope, improving detection accuracy by up to 8 new attack categories. Besides, our proposal can opportunistically detect the most reliable OS that should be used for the current environment behavior, improving by up to 8%, on average, the system accuracy when compared to a single OS approach, in the best case.**

***Index Terms*—Intrusion Detection, OS Diversity, SCADA, Machine Learning**

I. Introduction

Industrial Control Systems (ICS) are used for the monitoring and management of industrial processes, encompassing both hardware and software components [1]. Over the last years, modern ICS has become connected to a wide range of components, including Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCS), and even Programmable Logic Controllers (PLC) [2]. SCADA is of utmost importance in ICS; it responsible for the collection, monitoring, and handling of different industrial assets. Therefore, if an attacker can obtain remote privileged access to the SCADA server, she may cause significant economic losses or even the unavailability or destruction of industrial assets [3]. For instance, in 2015, a power plant in Ukraine was a target of a cyberattack that caused a countrywide blackout, affecting over 225 thousand clients [1].

The security of SCADA systems must be performed through a wide range of security mechanisms, including firewalls, air-gapped infrastructures, and Intrusion Detection Systems (IDS), used for the monitoring and detection of malicious activities in an environment [4]. In the last years, several approaches have been proposed for the detection task in IDS, through either *signature-based* or *behavior-based* techniques [5]. The prior refers to detection techniques wherein the attack pattern is previously known, thus, can be achieved through signature matching. However, SCADA is a frequent target of zero-day attacks, which cannot be detected by *signature-based* approaches, taking into account that such attacks will not have a matching signature. On the other hand, *behavior-based* approaches aim at modeling the system behavior for the detection of attacks, in which, in their majority, authors rely on machine learning (ML) techniques through pattern recognition approaches [6]. In such a scheme, a classification model, obtained through a training dataset, can be used for the detection of new attacks, as long as they behave similarly to the modeled threat [7].

Proposed ML approaches for intrusion detection tasks in general pursue higher detection accuracies, through custom-tailored ML algorithms, or more complex ML pipelines, e.g., further preprocessing tasks [8]. The range of attacks (detection surface) that can be detected by any ML algorithm is strictly related to the underlying set of used features. For instance, an IDS built for Linux Operating Systems (OS) is only able to detect attacks that affect the extracted feature set from the Linux OS. Even if a state-of-the-art ML algorithm is used – if its input features, tightly related to the environment – are not affected by an attack's occurrence, the attacker will be able to evade the detection mechanism [9]. This situation is even worse in the ICS, considering that attackers are highly motivated to evade detection mechanisms for-profit purposes [10] or due espionage activities [11].

This paper proposes a novel and reliable host-based intrusion detection model for SCADA systems that leverage OS diversity to increase the scope and accuracy of intrusion detection in a twofold implementation manner. First, our model continuously monitors host-based features for unreliable activities – that are not known to the underlying classification model, and the used OS. Second, our model leverages OS diversity for further increasing the detection surface, thus, increasing the system accuracy. The proposal insight is that higher detection accuracy can be achieved through OS diversity, rather than custom tailoring traditional ML classifiers.

In summary, the paper's contributions are:

- A publicly available and realistic testbed for the building of diverse IDS for SCADA. The built dataset is composed of over 100 legitimate clients and 19 different attack categories;
- Evaluation of traditional ML techniques for intrusion detection in SCADA. Our proposal results indicate that

978-1-7281-5414-5/20/$31.00 ©2020 IEEE

OS diversity can significantly improve detection accuracy for several attack categories;
- A novel, reliable intrusion detection model for SCADA based on OS diversity. Our proposed model can leverage OS diversity for increasing the overall system accuracy, while also detecting new attack categories to the underlying ML model;

II. Background

A. Supervisory Control and Data Acquisition

SCADA must be able to perform the collection and actuation on industrial assets through communication protocols such as MODBUS, DNP3, PROFINET, PROFIBUS, and OPC, in a master/slave scheme [10]. SCADA systems are high availability services, and its maintenance, for update purposes, for instance, becomes a challenge. SCADA security mechanisms must be as reliable as possible, taking into account that they must ensure the security of potentially outdated systems that are often the target of skilled and highly motivated attackers. SCADA system is commonly used in Smart Grid, an evolution of traditional energy systems has as its objective the generation, transmission, and distribution of electricity more autonomously and efficiently [12]

B. Host-based Intrusion Detection

In general, intrusion detection is achieved by four sequential modules, namely *data acquisition*, *feature extraction*, *detection*, and *alert*. The *data acquisition* module extracts the environment data for further inspection, e.g., the OS usage metrics such as used memory, number of processes, and connected users. *Feature extraction* module extracts behavioral features from the collected data, e.g., the number of the new process, and connected users in a given time window. Finally, the *classification* module applies a detection algorithm for the identification of intrusion attempts, e.g., a Machine Learning (ML) model classifies a given event as either normal or attack. The *alert* module then reports events classified attack.

The ML model is built from a training dataset, through a ML algorithm that extracts a behavioral model from such data. The ML model reliability relies on the input data used for training purposes, as well as the set of extracted features. In other words, if an attack does not affect the extracted feature values, or behaves similarly to the normal behavior, the ML model will misclassify it. Misclassification of events is typically measured through false-positive (FP) and false-negative (FN) rates. The FP rate denotes the ratio of normal events misclassified as an attack, while the FN rate denotes the ratio of attack events misclassified as normal.

III. Related Works

Authors often pursue Byzantine Fault Tolerance (BFT) to provide dependability and security in SCADA. However, SCADA is a highly complex system that can not be easily replicated as a state machine for BFT purposes. Nogueira et al. [13] have identified the properties that make the building of a BFT SCADA a challenge, which includes multiple entry points, multi-threading, non-deterministic timestamps, and asynchronous messages. As a result, the building of a SCADA applying system diversity is also an open challenge. The reason is its states must be replicated through the diverse platforms, for instance, OS and several server applications. Consequently, SCADA security is often pursued through traditional security mechanisms, such as firewalls and intrusion detection.

Intrusion detection for SCADA is, in general, performed through behavior-based approaches, applying a pattern recognition ML scheme. For instance, Shitharth S. et al. [14] propose a network-based intrusion detection scheme for SCADA applying neural networks and feature selection for network traffic classification. Their approach can improve detection accuracy through feature selection. However, their system accuracy is tightly coupled with the underlying feature set, while the detection of new attacks is not evaluated.

Leandros A. M. et al. [15] aims higher detection rates for intrusion detection in SCADA through an ensemble classifier coupled with feature selection. The authors were able to improve detection accuracy, but the detection of new attacks is not evaluated, and only a single subset of features is used for training and selection purposes.

Jiexin Z. et al. [16] proposed an intrusion detection scheme through telemetry and periodicity features analyzers. Their approach can increase detection accuracy through the evaluation of several subsets of features. The detection of new attacks is not evaluated. Besides, their approach relies on a single attack entry point, the SCADA server.

Rocio L. P. et al. [17] performed several preprocessing techniques and for a set of ML classification models to improve accuracy. The authors were able to improve detection accuracy through feature normalization and feature subset selection, while the detection of new attacks was not evaluated.

To the best of our knowledge, our work is the first approach that aims to reliable intrusion detection in SCADA applying OS diversity. Security through OS diversity is known and easy to replicate aspects that can be used for higher intrusion detection rates in SCADA [18].

IV. A Reliable Host-based Intrusion Detection Model for SCADA

In this section, we present the novel reliable host-based intrusion detection model for SCADA systems through OS diversity means. It consists of two main steps, namely *OS Behavior Classification* and *OS Pool-Selector*, that aim to improve SCADA security through OS diversity, further increasing the intrusion detection capability to detect new attacks.

The proposal, shown in Figure 1, considers a set of diverse OSs, which acts as the SCADA front-end. During execution, a single front-end is continuously used by the OS Behavior Classification module, e.g., a Windows OS acting as the SCADA front-end.

As our proposal considers *host-based* intrusion detection, an event can be represented as a set of OS-related usage metrics in a given time-window analysis, e.g., CPU usage in

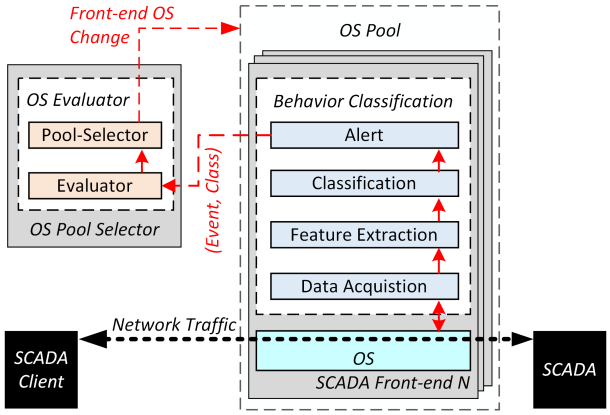

Fig. 1. Proposed reliable host-based intrusion detection model for SCADA through OS diversity.

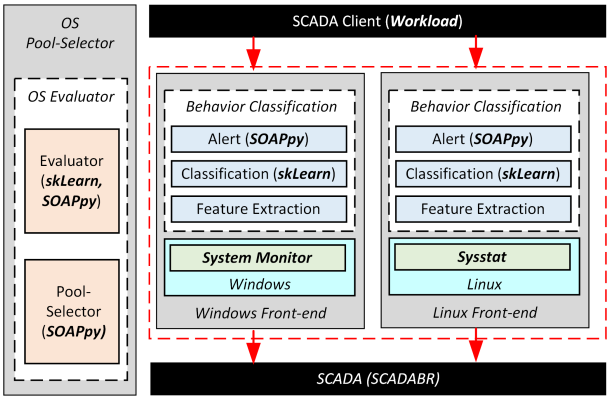

Fig. 2. Prototype architecture of the reliable host-based intrusion detection model for SCADA.

the last 10-s interval. The time window is classified by an ML model for finding anomalous/intrusion behavior. The event to be classified is sent by the OS Pool-Selector module, which evaluates the current environment behavior for selecting the most reliable OS.

The OS Pool-Selector module, in turn, aims at defining whether the currently used OS SCADA front-end can be reliably used for the detection of further intrusion attempts. The module applies an evaluator for the identification of unreliable (new to the ML model) behavior; for instance, it may evaluate the ML classification confidence for the current classified event.

Unreliable behavior is assumed to be intrusion attempts that can not be detected by the current SCADA OS front-end; thus, other OS must be used. Therefore, the system conservatively changes the SCADA OS front-end when unreliable events are detected in order to improve the overall system accuracy and reliability. The next subsections further detail the OS Behavior Classification and OS Pool-Selector modules.

A. OS Behavior Classification

Our assumption is that OS diversity is able to provide more accurate, more reliable and a higher detection scope. In other words, OS diversity improves the detection of other categories of attacks.

Running an intrusion detection mechanism that relies on SCADA diversity is not easy task. Our proposal is based on a SCADA front-end mechanism to facilitate the achievement of diversity. The proposal insight is that intrusion detection through diversity means can be achieved at the front-end level, instead of requiring the deployment of several OS with replicated SCADA servers. The rationale is SCADA front-end OS will also evaluate the network packets throughout the client/server communication; thus, the OS usage metrics will also be affected when the SCADA is under attack.

The OS Behavior Classification module is performed continuously by the current OS in execution as the SCADA front-end (SCADA Front-end N, Figure 1). The module is executed through the data acquisition stage, which periodically collects the OS usage metrics in a given period. Therefore, the module goal is to classify the current OS usage metrics (features' value), in a time window, as either normal or attack (intrusion).

The collected OS usage metrics values are used to compound a feature vector, which is then evaluated by the classification module. The final classification output and the feature vector is forwarded for the OS Pool-Selector module, for further inspection.

By using such an approach, our proposal can provide diverse OS metrics for the identification of attacks occurring in the SCADA server. Each OS front-end holds its ML model, built according to its OS metrics, achieving diversity in intrusion detection. Moreover, no additional changes are required to the SCADA server for the deployment of our proposal.

B. OS Pool-Selector

The *OS Pool-Selector* module goal is to define if the current SCADA front-end OS can detect the current environment behavior reliably. In other words, the module's goal is to establish which front-end OS must be currently used.

The module receives as input the current event classified by the OS Behavior Classification for further evaluation. Then, it evaluates if the classified event was reliably detected by the current front-end OS, defining whether the event is unknown to the underlying ML model or not, for instance, by evaluating the ML model classification confidence.

If a reliable event is found, the behavior is assumed to be known to the underlying ML model, and the current front-end OS remains unchanged. On the other hand, the module switches the current front-end OS in execution if an unreliable event is found (Front-end OS Change, Figure 1).

As a result, our proposal can continuously assess the current reliability of the SCADA front-end OS. Therefore, through the continuous evaluation of the current front-end performed classifications, the module establishes the most fitted front-end OS to be used for the current environment behavior, increasing the proposal reliability through diversity in intrusion detection.

V. Prototype

A proposal prototype was implemented and deployed in a distributed environment, as shown in Figure 2. The SCADA server was deployed through ScadaBR version 1.0CE. The proposed front-end mechanism was deployed on top of an Ubuntu virtual machine, which mirrors the network traffic – for this proof of concept (PoC), we did not switch the current OS, but consider both for comparison reasons. The SCADA clients access the SCADA server through the Ubuntu address, which then mirrors the request to the proper currently used front-end as defined by the OS Pool Selector module.

In the SCADA front-end, two OSs were used, a Windows 10 and an Ubuntu Linux 20.04. Both OSs were deployed as a virtual machine with four virtual cores and 8 GB of memory. The Windows and Linux OS metrics were collected through System Monitor and Sysstat, respectively. In total, 22 metrics are collected in Windows and 39 in Linux. The collected features are related to the OS usage statistics such as CPU usage time, number of processes, number of network packets sent/received, among others.

The features are summarized within a time-interval to compound a feature vector. For our evaluation, a 10-s window interval was used. The collected feature set is then classified by the ML algorithms as implemented through the SkLearn API version 0.23. The classified event and its features are sent to the OS Pool-Selector module through the SOAPpy web service API version 0.9.8. The OS Pool-Selector module evaluates the received event through SkLearn. If an unreliable event is found, the front-end updates the network traffic rules to forward further requests to another OS, transparently changing the SCADA OS for attack detection.

VI. Evaluation

The present evaluation focuses on answering three research questions: ($Q1$) *Does OS diversity aids at improving system detection accuracy?* ($Q2$) *Does the proposed model improve the overall detection accuracy?* ($Q3$) *What is the front-end frequency change over time?*

The next subsections describe how do we build the model and testbed, and how does it perform when facing new SCADA targeted attacks.

A. Testbed

Our proposal relies on a set of a different OS to provide reliability to SCADA. However, the majority of SCADA related intrusion datasets in the literature are outdated, unrealistic, and specific to a single testbed setting. Therefore, for the proper evaluation of our proposal, we have assembled a testbed containing diverse and realistic behavior from SCADA production environments.

In our testbed, a SCADA server is deployed through the SCADABR version 1.0CE, with several popular services typically reported in the literature [19]. The built testbed comprises two possible behaviors, namely normal and attack.

TABLE I
Windows and Linux OS front-end testbed data.

Behavior (Tool)	Testbed			
	Windows		Linux	
	Net. Pkt.	Exec. Time (m)	Net. Pkt.	Exec. Time (m)
Normal (Workload)	5.1B	5760	5.1B	5760
Auth Crawler (Acunetix)	77k	313	77k	311
Nonauth Crawler (Acunetix)	128k	184	129k	182
SQL Injection (Acunetix)	13k	212	13k	210
XSS (Acunetix)	6k	1028	6k	1027
Code Injection (Arachni)	17k	392	17k	391
DoSAllRegisters (Smod)	58k	153	58k	152
Get Functions (Smod)	2k	133	2k	133
Scanner (ModFuzzer)	0.4k	4	0.4k	4
DumpScan (ModFuzzer)	1k	4	1k	4
Portscan (Nmap)	140k	33	140k	33
Modbus Scan (PLCScan)	3k	22	3k	21
Basic scan (Nessus)	45k	52	45k	52
Advanced scan (Nessus)	35k	33	36k	34
Default scan (Nexpose)	150k	56	151k	56
Advanced scan (Nexpose)	38k	86	39k	87
SQL Injection (Arachni)	4k	70	4k	69
Fuzzing (Smod)	320k	20	320k	20
Scanner (Smod)	2k	113	3k	112
DoSAllCoils (Smod)	296k	164	296k	160

The normal behavior is generated by 100 client machines that continuously use HTTP, HTTPS, SSH, SMTP, and MODBUS protocol communication with the SCADA server. Each normal service communication varies the client throughput (35%, 70%, and 100% gigabit throughput), request frequency ($0sec$ to $4sec$), and requested content (1000 possible service content for each protocol).

Each used protocol was generated through well-known workload tools, while the ScadaBR server properly reply each received request. As a result, it mimics the highly variable behavior of legitimate SCADA clients in real-world environments.

For the attack behavior generation, several well-known and publicly available tools were used, taking into account the commonly observed attacks in both web applications and SCADA servers, considering that SCADA is, in general, exposed to the Internet. In total, 19 attacker machines periodically generate application and network-level attacks targeted to the SCADA server. Similarly, the attack behavior varied attack frequency and throughput.

The testbed used the same settings and tools used for prototype development (Figure 2). Therefore, two testbed executions were performed for the proper data collection of OS diversity. One execution used a Windows OS as a front-end for both normal and attacker machines, and the other uses the Linux OS as front-end.

Table I shows the amount of generated network packets and execution time for each attack in our testbed for both Windows and Linux scenarios, where each execution lasted 96 hours. It is important to note that, as implemented by our prototype (described in section V), one instance for classification is generated at every 10-s time interval for both Windows and Linux OS front-ends.

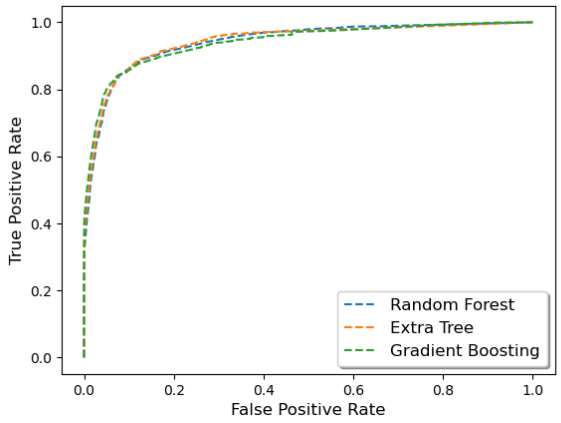

Fig. 3. ROC curve for three classifiers in Windows OS test data. It is possible to note similar results despite using different machine learning algorithms.

TABLE II
CLASSIFICATION PERFORMANCE WITH RANDOM FOREST CLASSIFIER OBTAINED BY WINDOWS, LINUX AND OUR PROPOSAL.

Behavior (Tool)	Accuracy (%)			
			Proposal	
	Windows	Linux	Worst	Best
Normal (Workload)	91.40	**98.79**	91.40	98.79
Auth Crawler (Acunetix)	85.00	**100.00**	93.76	100.00
Nonauth Crawler (Acunetix)	**86.74**	83.00	86.72	86.74
SQL Injection (Acunetix)	84.66	**100.00**	93.66	100.00
XSS (Acunetix)	**83.74**	77.98	83.73	83.74
Code Injection (Arachni)	**90.03**	83.50	90.01	90.03
DoSAllRegisters (Smod)	84.05	**96.83**	92.07	96.83
Get Functions (Smod)	85.71	**100.00**	94.67	100.00
Scanner (ModFuzzer)	**93.75**	75.00	93.74	93.75
DumpScan (ModFuzzer)	**94.11**	54.54	94.07	94.11
Portscan (nmap)	88.41	**98.03**	98.01	98.03
Modbus Scan (PLCScan)	**81.19**	55.29	81.13	81.19
Basic scan (Nessus)	72.16	**82.48**	81.61	82.48
Advanced scan (Nessus)	79.05	**95.65**	93.85	95.65
Default scan (Nexpose)	70.74	**90.15**	90.10	90.15
Avg. System Accuracy	84.72	86.08	**90.57**	**92.77**

B. Model Building

The built datasets were used for the building of two OS-related ML classification models, for Windows and Linux OSs. The testbed data was split into train, test, and validation data. The train and test datasets were composed by 50% of the total normal events, and 5 out of 19 attack categories (*Advanced scan (Nexpose)*, *SQL Injection (Arachni)*, *Fuzzing (Smod)*, *Scanner (Smod)*, and *DoSAllCoils (Smod)*, Table I), wherein 70% of such data was used for training and the remaining 30% for testing.

The remaining 50% of normal traffic and 14 attack categories are used for the final ML model validation. By applying such an approach, we can reproduce the production environment behavior, wherein the ML model will encounter new (unknown) attack behavior. The evaluated classification models were implemented through the SKLearn API version 0.23.

Three widely used classification models were evaluated, a Random Forest and a Gradient Boosting both with 100 decision trees as their base learners and an Extra Tree classifier. The classifiers were trained through train and test datasets, while the final accuracy is reported through the validation dataset. A random undersampling without a replacement algorithm was applied in the training data to remove the unbalance between the classes.

C. On OS Diversity for Reliable Intrusion Detection

The first experiment relates to question $Q1$ and evaluates the classifier performance for the classification of new attack categories through different OS feature sets. Figure 3 shows the Receiver Operating Characteristics (ROC) curve for the three evaluated ML models in the Windows OS test dataset. It is possible to note that the evaluated classifiers presented similar accuracy rates concerning their underlying OS feature sets; thus, ML classification accuracies are strictly dependent on the used feature set.

To evaluate the OS diversity impact on classification accuracy, we have compared the reported accuracy on Windows, and Linux feature sets regarding the validation dataset. Recalling that the validation dataset is made of new attack categories behavior, i.e., behavior not used at the training phase.

Table II (*Windows* and *Linux* columns) shows the obtained accuracy for the Random Forest classifier for each attack category and OS feature set. Differently from the previous analysis, the classification performance varies significantly according to the used OS feature set. It is possible to note that the Linux classifier outperforms the Windows approach for the detection of normal behavior.

Concerning the classification of new attack categories, the Windows-related ML model can provide higher detection accuracies for six attack categories, while the Linux-related model achieves higher detection rates for eight attack categories. Therefore, OS diversity for intrusion detection aids in improving the overall intrusion detection rate for new attacks.

The second experiment relates to question $Q2$ and aims at evaluating the *OS Pool-Selector* approach for accuracy improvement through OS diversity. We implemented a Class-Related-Threshold (CRT) approach for establishing whether the current OS in use should be maintained or changed. In other words, the proposal evaluator (*Evaluator*, Figure 1) maintains or switches the current OS according to its ML model classification confidence value. For instance, the Random Forest classifier outputs a confidence value according to the ratio of individual trees that assigned the event assigned label. In this PoC, we do not need to switch the current OS, because the traffic is mirrored.

The OS switch frequency must be performed according to the administrator discretion; for our tests, we have selected a 0.9 confidence value threshold. Thus, performed classifications with less than 0.9 confidence value trigger a different OS front-end classification mechanism.

Table II (*Proposal* columns) shows the obtained accuracy

using our proposal, considering both the worst and best scenarios while proactively switching the current OS front-end according to its classification confidence.

The worst scenario is found when the current OS in usage provides the lowest detection accuracy for the current attack; the average system accuracy on 84.72%. Hence, the used CRT approach ensures the accuracy level through the defined threshold (0.9 for our experiments). On the other hand, the best scenario is found when the current OS provides the highest detection accuracy for the current attack, the average system accuracy on 92.77%.

It is possible to note that our proposal is always able to outperform in about 6% the OS that obtained the lowest accuracy (worst scenario, Avg. system Acurracy = 90.57 - 84.72 = 5.85%), either Linux or Windows. Similarly, it is also possible to reach the same level of the best accuracy rates to those obtained from the best OS according to the current classified attack category (best scenario, Avg. system Acurracy = 92.77 - 84.72 = 8.05%). In other words, the proposal can leverage the accuracy improvement obtained from OS diversity to proactively identify and switch the current used OS to increase the overall attack classification accuracy, as shown in Table II (*Avg. System Accuracy*).

The third and final question $Q3$ is regarding the OS change frequency in our model. Such property is highly dependent on current environment behavior. For instance, consider the *Auth Crawler (Acunetix)* attack category, shown in Table II. If the Windows OS is currently being used as a front-end, the proposal will change it as soon as it classifies an event with less than 0.9 confidence. Afterward, the Linux OS will not be switched, taking into account that it will always perform reliable classifications (100% accuracy, as shown in Table II). On the other hand, considering the *Nonauth Crawler (Acunetix)* attack, regardless of the OS being used, several changes might occur. It is because both OSs may perform unreliable classifications, as observed through their obtained accuracy rates ($\sim$85%). One may use the OS frequency switch as an indication that a new attack is occurring, while low accuracy is being achieved for its classification, demanding further administrator investigation

VII. Conclusion and Future Work

Our proposal is able to achieve higher accuracy rates through OS diversity in intrusion detection and the continuous evaluation of the system reliability, while not affecting or requiring changes in the SCADA system, operating transparently. The performed evaluations have shown that feature set diversity, through OS diversity, aids at improving the detection of the new attack behavior. Moreover, our proposal was able to proactively identify and switch the current OS that should be used to provide reliability while detecting new attack behavior.

As future work, we will deploy our proposed model in Software-Defined Networks for higher transparency, leverage higher OS diversity, and perform the detection through an ensemble of OS diversity feature sets.

Acknowledgment

This work was partially supported by the CNPq (National Council for Scientific and Technological Development) grant 315322/2018-7.

References

[1] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the ukrainian power grid," *Document*, Mar. 2016. [Online]. Available: https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf

[2] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ICS) security," Tech. Rep., Jun. 2011.

[3] V. Abreu, A. O. Santin, E. K. Viegas, and V. V. Cogo, "Identity and access management for IoT in smart grid," in *Advanced Information Networking and Applications*. Springer International Publishing, 2020, pp. 1215–1226.

[4] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, "Guide to industrial control systems (ICS) security," Tech. Rep., Jun. 2015. [Online]. Available: https://doi.org/10.6028/nist.sp.800-82r2

[5] E. K. Viegas, A. O. Santin, V. V. Cogo, and V. Abreu, "Facing the unknown: A stream learning intrusion detection system for reliable model updates," in *Advanced Information Networking and Applications*. Springer International Publishing, 2020, pp. 898–909.

[6] E. Kakihata, H. Sapia, R. Oikawa, D. Pereira, J. Papa, V. Alburquerque, and F. Silva, "Intrusion detection system based on flows using machine learning algorithms," *IEEE Latin America Transactions*, vol. 15, no. 10, pp. 1988–1993, Oct. 2017.

[7] E. Viegas, A. O. Santin, A. Franca, R. Jasinski, V. A. Pedroni, and L. S. Oliveira, "Towards an energy-efficient anomaly-based intrusion detection engine for embedded systems," *IEEE Transactions on Computers*, vol. 66, no. 1, pp. 163–177, Jan. 2017.

[8] C. Vicentini, A. Santin, E. Viegas, and V. Abreu, "A machine learning auditing model for detection of multi-tenancy issues within tenant domain," in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, May 2018.

[9] E. K. Viegas, A. O. Santin, V. V. Cogo, and V. Abreu, "A reliable semi-supervised intrusion detection model: One year of network traffic anomalies," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. IEEE, Jun. 2020.

[10] A. Zimba, Z. Wang, and H. Chen, "Multi-stage crypto ransomware attacks: A new emerging cyber threat to critical infrastructure and industrial control systems," *ICT Express*, vol. 4, no. 1, pp. 14–18, Mar. 2018.

[11] Danish Saleem and C. Carter, "Certification procedures for data and communications security of distributed energy resources," 2019.

[12] X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid — the new and improved power grid: A survey," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 944–980, 2012.

[13] A. Nogueira, M. Garcia, A. Bessani, and N. Neves, "On the challenges of building a BFT SCADA," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, Jun. 2018.

[14] S. S and P. W. D, "An enhanced optimization based algorithm for intrusion detection in SCADA network," *Computers & Security*, vol. 70, pp. 16–26, Sep. 2017.

[15] L. A. Maglaras, J. Jiang, and T. J. Cruz, "Combining ensemble methods and social network metrics for improving accuracy of OCSVM on intrusion detection in SCADA systems," *Journal of Information Security and Applications*, vol. 30, pp. 15–26, Oct. 2016.

[16] J. Zhang, S. Gan, X. Liu, and P. Zhu, "Intrusion detection in SCADA systems by traffic periodicity and telemetry analysis," in *2016 IEEE Symposium on Computers and Communication (ISCC)*. IEEE, Jun. 2016.

[17] R. L. Perez, F. Adamsky, R. Soua, and T. Engel, "Machine learning for reliable network attack detection in SCADA systems," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications (TrustCom)*. IEEE, Aug. 2018.

[18] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro, "Analysis of operating system diversity for intrusion tolerance," *Software: Practice and Experience*, vol. 44, no. 6, pp. 735–770, Jan. 2013.

[19] S. Ghosh and S. Sampalli, "A survey of security in SCADA networks: Current issues and future challenges," *IEEE Access*, vol. 7, pp. 135 812–135 831, 2019.