Authenticating IDS Autoencoders Using Multipath Neural Networks

Raphaël Larsen, Marc-Oliver Pahl, Gouenou Coatrieux

Authenticating IDS Autoencoders Using Multipath Neural Networks

Raphaël M.J.I. Larsen
Institut Mines-Télécom Atlantique, Rennes, France
Email: raphael-larsen[at]hotmail.fr

Marc-Oliver Pahl
Institut Mines-Télécom Atlantique, Rennes, France
Email: marc-oliver.pahl[at]imt-atlantique.fr

Gouenou Coatrieux
Institut Mines-Télécom Atlantique, Brest, France
Email: gouenou.coatrieux[at]imt-atlantique.fr

***Abstract*—An Intrusion Detection System (IDS) is a core element for securing critical systems. An IDS can use signatures of known attacks, or an anomaly detection model for detecting unknown attacks. Attacking an IDS is often the entry point of an attack against a critical system. Consequently, the security of IDSs themselves is imperative. To secure model-based IDSs, we propose a method to authenticate the anomaly detection model. The anomaly detection model is an autoencoder for which we only have access to input-output pairs. Inputs consist of time windows of values from sensors and actuators of an Industrial Control System. Our method is based on a multipath Neural Network (NN) classifier, a newly proposed deep learning technique. The idea is to characterize errors of an IDS's autoencoder by using a multipath NN's confidence measure c. We use the Wilcoxon-Mann-Whitney (WMW) test to detect a change in the distribution of the summary variable c, indicating that the autoencoder is not working properly. We compare our method to two baselines. They consist in using other summary variables for the WMW test. We assess the performance of these three methods using simulated data. Among others, our analysis shows that: 1) both baselines are oblivious to some autoencoder spoofing attacks while 2) the WMW test on a multipath NN's confidence measure enables detecting eventually any autoencoder spoofing attack.**

I. Introduction

Intrusion Detection Systems (IDSs) are the primary security tools for Industrial Control Systems (ICSs) [17]. IDSs date back to the early 80's. They are systems whose objective is to detect malicious activity and policy violations regardless of "*whether they are initiated by outsiders who attempt to break into the system or insiders who attempt to misuse the privileges of their accounts*" [5]. While early IDSs were based on signatures of known attacks only, recent solutions rely on anomaly detection models that also detect unknown attacks.

The used models are typically unsupervised: they do not need examples to be trained to detect anomalies, such as functional failure or attacks. This is important as attack examples are rare, especially for Cyber-Physical Systems (CPSs). ICSs often control several CPSs. At the same time, CPSs are more likely to be faced with unknown attacks than Information and Communications Technology systems [13].

This work was supported by the Cyber CNI Chair and IMT Atlantique, Bretagne-Pays de la Loire.

Anomaly detection for ICS follows three directions [3]: 1) protocol, 2) network and 3) payloads, i.e., sensors measurements, automata commands and actuators responses, referred to as sensor data. Protocol-based anomaly detection identifies anomalies on the basis of deviation from normal packet data and protocol rules. For network-based anomaly detection, a sequence of network packets is used, modeling the communication relationships between devices. For payload-based anomaly detection, it is the modeling of the physical relationships between devices as well as the normal range of system behavior that are considered.

The use of Deep Learning techniques to build an anomaly detection model for the security of CPSs, including ICSs, has increased in recent years [2], [4], [18], [13]. The anomaly detection method used for monitoring an ICS is a critical tool that might be targeted by attackers. The security of IDSs themselves matters.

Yet, [15] reports that security of most of IDSs is low and only one paper [16] that "*concerns itself with the nature of attacks against intrusion detection systems themselves*" [15] is cited. While this survey dates from 1998, a decade later, in 2009, [12] stated that "*more significant efforts should be done to improve intrusion detection technology in this aspect* [security]". In the case of Deep Learning techniques, this concern is even more relevant since they are recent techniques whose security is not completely understood yet. The first survey on attacks against IDS dates back from 2013 [9]. It distinguishes six attacks against IDS, including Response Hijacking—"*a pattern is crafted to generate an incorrect alert description and mislead the IDS response mechanism (either automatic or performed by a security operator)*" [9]—which our study focuses on. In 2016, another survey [10], about the evasion resilience of IDSs' machine learning methods, shows that the security aspect of many of their anomaly detection models is an oversight. To the best of our knowledge, there is no method that tackles the security of an anomaly detection model used by an IDS. This study is restricted to the security of payload-based anomaly detection models monitoring an ICS. The research question is how to authenticate these models.

In this paper, the anomaly detection method used by the IDS to monitor the ICS physical process is based on an

autoencoder. We propose a method to verify that the genuine autoencoder is working by probing its outputs. Otherwise, we assume an autoencoder spoofing attack. Our method bases on a statistical test on the confidence measure of a multipath Neural Network (NN) classifier [14]. If the test indicates a likely change of distribution of a multipath NN's confidence measure, an alert is raised. Results show that our method can detect any autoencoder spoofing attack with an arbitrary False Positive Rate (FPR) fixed in advance, yet the lower the FPR, the longer the detection.

In the rest of the paper, we first present in Section II necessary background knowledge for understanding the paper which comprises 1D-Convolutional Neural Network (CNN) autoencoder, multipath NN and Wilcoxon-Mann-Whitney test. Section III presents our method. Section IV presents the dataset used for evaluation, details two baselines to be compared to our method, exposes and discusses the results.

II. Background and assumptions

A. Motivated assumptions

We assume that an IDS anomaly detection model gets an input x (sensor data) and outputs a security signal y based on which the security team can take a decision. For instance, if the anomaly detection model is an autoencoder, the security signal y is the reconstruction of x and the security team decide whether there is an anomaly based on the differences between x and y. Too much difference suggests an anomaly in sensor data x. We would like to be sure that the security signal comes from the genuine model and no one is spoofing the model, by providing a security signal $y_{attacker}$ trusted to be from the IDS, to prevent the security team to be aware of a threat.

A trivial solution is to replicate the autoencoder of the IDS so that the security team just has to compare the output of their autoencoder y_{team} to the one of the monitoring autoencoder y. If $y \neq y_{team}$, then the monitoring autoencoder is spoofed. However, this solution adds a course of action for a malicious actors who now has two ways to learn about the anomaly detection model: through the IDS and through the replicated autoencoder. Not to mention that this implies that the security team has access to the anomaly detection model, including its architecture and weights in case a neural network is used.

Another solution is to add classical security tools to authenticate the IDS's anomaly detection model. For example, one could use a cryptographic message authentication code $tag_{IDS}(y)$, also known as tag, so that with the proper secret key, they can verify that $tag_{IDS}(y) = tag_{team}(y)$. But solutions like this require computational resources to be added to the IDS to get $tag_{IDS}(y)$. Furthermore, if an attack is committed by an insider, by simply replacing the monitoring autoencoder of the IDS by another, one cannot detect this attack by the means of tags anymore.

In contrast, the solution we propose does not imply access to the IDS anomaly detection model nor additional computational resources. Instead its constraints are of another kind, it requires several input-output pairs (x,y) to authenticate the IDS anomaly detection model and its authentication is not strict but is rather given by a level of confidence. In other words, a security operator can only probe the output of the autoencoder, thus adding a layer of security, the authentication of the anomaly detection model, without compromising the previous one, the anomaly detection itself.

B. 1D-CNN autoencoder

In this paper, we consider several kinds of NNs. In particular, the anomaly detection model of the IDS is a 1D-CNN autoencoder.

NNs consist of several layers, each composed of several neurons. Given couples of input and target values, NNs are trained to predict the target from a particular input using a loss function. The loss function is applied on the output layer and a backpropagation algorithm enables changing the weights that link neurons between two different layers.

Autoencoders are NNs whose input and output have the same number of dimensions and they are trained to reconstruct their inputs. Because of some constraints on the hidden layers, such as dimension reduction, an autoencoder better reconstructs points close to the training distribution than points far from it. In an anomaly detection setting, the reconstruction error is used as an anomaly score. If the average error of reconstruction of an input is above a certain threshold, the input is considered dissimilar enough to those of the training set to raise an alert.

Autoencoders can be of the form of a Convolutional Neural Network (CNN). CNNs are NNs with drastically fewer weights than an FC (Fully Connected) NN. More specifically, a nD-CNN is composed of convolutional layers and pooling layers. It takes inputs with n spatial dimensions. Typically, a time window has one spatial dimension ($n = 1$), generally called temporal dimension. An image has two spatial dimensions ($n = 2$), height and width. CNNs pooling layers change the size (i.e., number of features) of the next layer without adding learnable parameters. A convolutional layer consists of sets of filters that are sets of kernels (tensors of order n). They apply an operation similar to the well-known convolution, namely the cross-correlation, so as to capture spatial patterns along n dimensions. In a layer after the input layer, the channel dimension corresponds to the number of filters that extract potentially different patterns. The channel dimension of the input layer depends on the input shape. For example, in a 1D-CNN, for a k-dimensional time series, there are k input channels. Details about CNNs can be found in [11].

In this paper, the IDS's anomaly detection model is a 1D-CNN autoencoder. Its inputs are time windows of a k-dimensional time series of physical process data, with k the number of sensors and actuators of the observed ICS.

C. Multipath Neural Network

An important building block of our method to authenticate a 1D-CNN autoencoder is the multipath NN model.

A multipath NN is a NN with at least one layer composed of Dissector Layers (Diss-Layers) [14]. A Diss-Layer is a computational unit whose output has the same number d of

dimensions as the input and that is defined by a simple relation that uses $3 \times d + 1$ learnable parameters. This computational unit's sum of activations is a surrogate for the left-hand side of the equation of a hyperplane $\sum_{i=1}^{d} a_i \times x_i + a_0 = 0$ so as to benefit from a property of supervised NN classifiers shown in [7]. This property is that the level of linear separability increases monotonically when we go to deeper layers. Diss-Layers are typically in the last layers.

Several Diss-Layers compose a Hyper-Neuron. A multipath NN classifier forces, during the training phase, the information to go through the dedicated Hyper-Neuron depending on the input class. During inference, it provides a confidence measure that verifies that the information goes through the right Hyper-Neuron. Its confidence measure is shown to be robust to transformations of the input distribution that mainly occur in non-discriminative features, i.e., features not responsible for the class membership.

In this paper, a multipath NN's confidence measure characterizes an autoencoder's errors using a statistical test.

D. Wilcoxon-Mann-Whitney test

The second building block of our method to authenticate a 1D-CNN autoencoder is the Wilcoxon-Mann-Whitney (WMW) test. This test is distribution-free, i.e., data are not assumed to follow a particular family of distributions, which makes our method practical for any ICS.

The WMW test [6], [8] allows to check whether two samples $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$ are from the same distribution. The assumption of this test is that the observations from both samples are independent of each other. The null hypothesis H_0 is that the distributions are the same, and the alternative hypothesis H_1 considered herein is that the distributions are different. The statistic U of this test is the minimum between the number of times a Y precedes a X, U_1 and the number of times a X precedes a y, $U_2 = nm - U_1$, in the ordered sequence composed of X's and Y's. More formally, $U = \min(U_1, U_2)$ and, with $\mathbb{1}$ the indicator function:

$$U_1 = \sum_{i=1}^{n} \sum_{j=1}^{m} \mathbb{1}_{X_i > Y_j} + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} \mathbb{1}_{X_i = Y_j}$$

The test is consistent—i.e., the probability to reject H_0 when H_1 is true converges towards 1 when $n, m \to +\infty$—only if $\mathbb{P}(X > Y \,|\, H_1) \neq \mathbb{P}(Y > X \,|\, H_1)$, with $\mathbb{P}$ the notation for probability. For samples of size greater than 20, U_1 and U_2 are approximately normally distributed, so one can decide an appropriate acceptable False Positive Rate (FPR) α, such that H_0 is rejected whenever the p-value—probability under H_0 that $|U| > |u|$, with U the statistic and u the observation of this statistic—is lower than α.

In this paper, we propose to perform the WMW test on the confidence measure of a multipath NN. A sample S_1 of confidence measures known to be derived from the genuine autoencoder of the IDS serves as a reference to test if a second sample S_2 is from the same distribution.

III. Authentication of a 1D-CNN autoencoder

Our method to authenticate an 1D-CNN autoencoder that monitors the time series of an ICS's sensors and actuators values relies on three steps. The first step consists in choosing moments in the ICS cycle that serve to characterize the autoencoder based on time windows corresponding to these moments. The second step is the training of a multipath NN classifier that must predict the moment when given the matrix of errors of the autoencoder that processed the corresponding time window. The third step is a hypothesis testing that aims at verifying that the distribution of the multipath NN's confidence measure has not changed.

A. Preprocessing of time series of sensors and actuators values

Before explaining the first step in detail, we need to introduce some notation. We consider an ICS composed of k sensors/actuators, now referred to as items. The ICS's physical process data form a k-dimensional time series, i.e., a time series that yields k values at each time step. We denote $R^c_{i,j}$ the value of the i-th item at the j-th time step within the c^{th} ICS cycle. We assume that the duration between two time steps is always the same. For the sake of simplicity, we do not consider a particular cycle instance c anymore but the random variable $R_{i,j}$ instead, i.e., the function associating $R^c_{i,j}$ to c.

Now, we can explain the first step, i.e., choosing moments in the ICS cycle. Let us first assume that the ICS executes the same sequence of actions each cycle. In this case, the time window of the n-th moment of interest can simply be $R^n = ((R_{i,j})_{1 \leq i \leq k})_{\lfloor p_n \times L/l \rfloor \leq j < \lfloor p_n \times L/l \rfloor + s}$, with L the variable length of the cycle, $l = \min(L)$, p_n positions parameters and s the size of time windows. In other words, the cycle is simply divided into windows whose positions $\lfloor p_n \times L/l \rfloor$ are proportional to the cycle length as depicted in Figure 1. If the size of the time windows is big enough with respect to the cycle variability, one can assume that a moment defined as in Section III-A will correspond to a same set of ICS's high-level actions from one cycle to another.

High-level actions are actions that an ICS is programmed to perform in its physical environment. For example, drilling an object or the manipulation of an object by a robotic arm. In other words, results of the working of an ICS at the human level. In this way, time windows from the same moment in different cycles are similar enough for a classifier to predict the moment given a corresponding time window. It enables training a classifier to predict the moment given the autoencoder's matrix of errors of reconstruction of a corresponding time window. This is needed for the next step.

In general, an ICS does not necessarily executes the same sequence of actions in each cycle. As an example, a car manufacturing system equipped of a single production line can produce vehicles with different features. The ICS will conduct different sequences of actions depending on the car to produce. In this case, the system's knowledge is necessary to define moments that correspond to some high-level sets of actions. However, the purpose is the same as before, that is one must

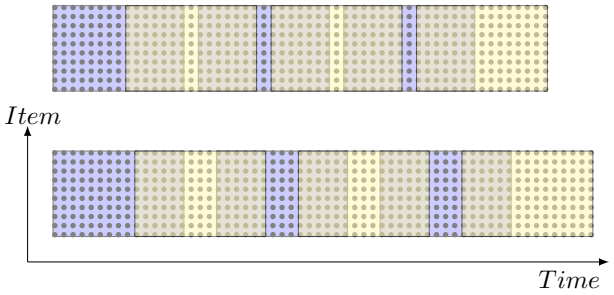

Fig. 1. Two ICS cycles with fixed size time windows from the time series of sensors/actuators values (items) represented by gray dots. Time windows are linked to 6 distinct moments whose positions in the cycle are proportional to its length. Due to the ICS's working variability, the cycle length is variable.

be able to predict the moment given the autoencoder's matrix of errors of reconstruction of a corresponding time window. Once this verification done, one can proceed to the next step.

B. Training of a multipath NN classifier

The second step, after having specified moments in the ICS cycle as explained in the previous section, consists in training a multipath NN to predict the moment based on the matrix of errors—element-wise squared difference between the input and the output—of the monitoring autoencoder that processed a time window corresponding to this moment. This NN comes with a confidence measure that is robust to some transformations of the input (cf. Section II-C). The idea is to characterize errors of an IDS's autoencoder thanks to a multipath NN's confidence measure. Once m moments have been chosen, a multipath NN classifier is defined such that its last layer is a fully connected layer with m output features and its first layers are convolutional layers. Then, one needs a balanced—about the same number of examples for each class/moment—set S of matrices of reconstruction errors from the monitoring autoencoder that are normalized with the min-max normalization. To this set of size $|S|$ are added $\lfloor |S|/m \rfloor$ random matrices from the unit hypercube (cf. [14]): consider a matrix as a vector and draw each dimension's value from the uniform distribution between 0 and 1. The set obtained is the training set of the multipath NN. Once the training set defined, one uses the multipath NN's loss detailed in [14] that ensures that the multipath NN's confidence measure is properly learned. This confidence measure is simply the softmax output's max value multiplied by a term which allows verifying that the path of the information leading to the prediction is the expected one.

C. The Wilcoxon-Mann-Whitney test on the multipath NN's confidence measure

The third step is a Wilcoxon-Mann-Whitney (WMW) test on the confidence measures of the multipath NN. One has to decide the size of the tested sample and the size of the sample of reference. The latter is the confidence measures' sample arising during the normal operation of the ICS (without attack or dysfunction) and with the genuine autoencoder. Typically, the sample of reference is large (≥ 100), and to be able to use any FPR α (cf. Section II-D) manageable for the security team, the size of tested samples has to be greater than 20 as explained in Section II-D. The more subtle is the attack, the larger has to be the tested samples. Thus, if the multipath NN is used online, one can perform several tests on increasingly larger samples to rapidly detect rather obvious attacks and eventually detect more subtle ones.

One has to build a sample of reference in the same way as the set S from Section III-B—but different from S—except that one has to use the same normalization parameters as before (cf. Section III-B) instead of recomputing them based on the new matrices. The same holds for tested samples, but, of course, one does not know whether matrices of errors are from the genuine autoencoder. An alert is raised whenever the WMW test rejects the null hypothesis.

IV. Experimentation

The experimentation consists in testing our method in several settings, that is with different 1D-CNN autoencoders and multipath NNs, as well as two baselines introduced in Section IV-A. After presenting the data in Section IV-B and the results in Section IV-C, we compare the ability of each method to distinguish normal samples from abnormal ones with Receiver Operating Characteristic (ROC) curves, which plot True Positive Rate (TPR) against FPR.

A. Baselines and hyperparameters

Our method aims to characterize an autoencoder's reconstruction errors by providing to the WMW test a confidence measure on its ability to retrieve the moment within the ICS cycle so as to detect spoofing attacks. One can ask if there is not a better variable to summarize an autoencoder's errors. The first baseline is the use the autoencoder's mean squared error for the WMW test. The second baseline involves the confidence measure of a classical NN classifier, that is its softmax's maximum value, as summary variable for the test.

The architectures of the classical NNs and of the multipath NNs, all trained in batches of size 16 and in 20 epochs, are:

- Classical NN classifier

$InputLayer$(15,15) $\rightarrow$ $Conv$(15,19) $\rightarrow$ Up(4)(60,19)
$\rightarrow$ $Down$(5)(12,19) $\rightarrow$ $Conv$(12,22) $\rightarrow$ Up(4)(48,22)
$\rightarrow$ $Down$(5)(10,2) $\rightarrow$ $Conv$(10,24) $\rightarrow$ Up(4)(40,24)
$\rightarrow$ $Down$(5)(8,24) $\rightarrow$ $Conv$(8,25) $\rightarrow$ $Flat$(200) $\rightarrow$ $Dense$(5)

- multipath NN classifier (simplest form: one Diss-Layer per dedicated Hyper-Neuron)

$InputLayer$(15,15) $\rightarrow$ $Conv$(15,19) $\rightarrow$ Up(4)(60,19)
$\rightarrow$ $Down$(5)(12,19) $\rightarrow$ $Conv$(12,22) $\rightarrow$ Up(4)(48,22)
$\rightarrow$ $Down$(5)(10,2) $\rightarrow$ $Conv$(10,24) $\rightarrow$ Up(4)(40,24)
$\rightarrow$ $Down$(5)(8,24) $\rightarrow$ $Conv$(8,25) $\rightarrow$ $Flat$(200)
$\rightarrow$ $Dense$(100) $\rightarrow$ $5DissLayers$(100) $\rightarrow$ $Dense$(5)

A couple or a number after layers $Up(n)$, $Down(m)$, $Conv$, $Dense$, $Flat$ or $5DissLayers$ indicates the layer's output shape. $Up(n)$ repeats data n times (for example, $Up(2)([1,2,3]) = [1,1,2,2,3,3]$), and $Down(m)$ operates a max-pooling with strides m and pool size 2 (for example,

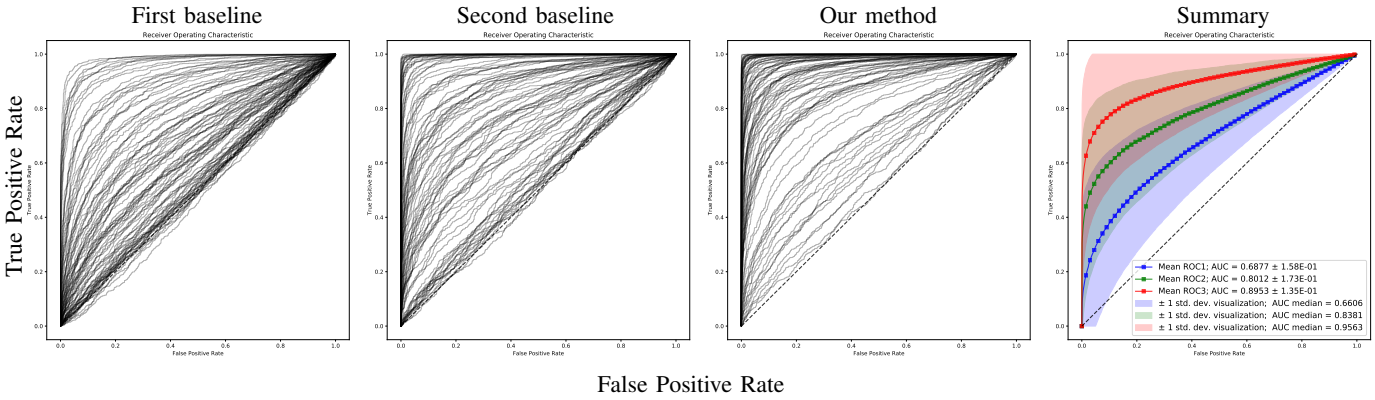

Fig. 2. ROC curves from 1 - p-values of the WMW test on the three summary variables (Section IV-A) for samples of size 25 tested against a reference sample of size 1000. First 3 graphs: 90 ROC curves in each, details in Fig. 4 and 5. Fourth graph: mean curves (ROC1/2: 1st/2nd baseline, ROC3: our method).

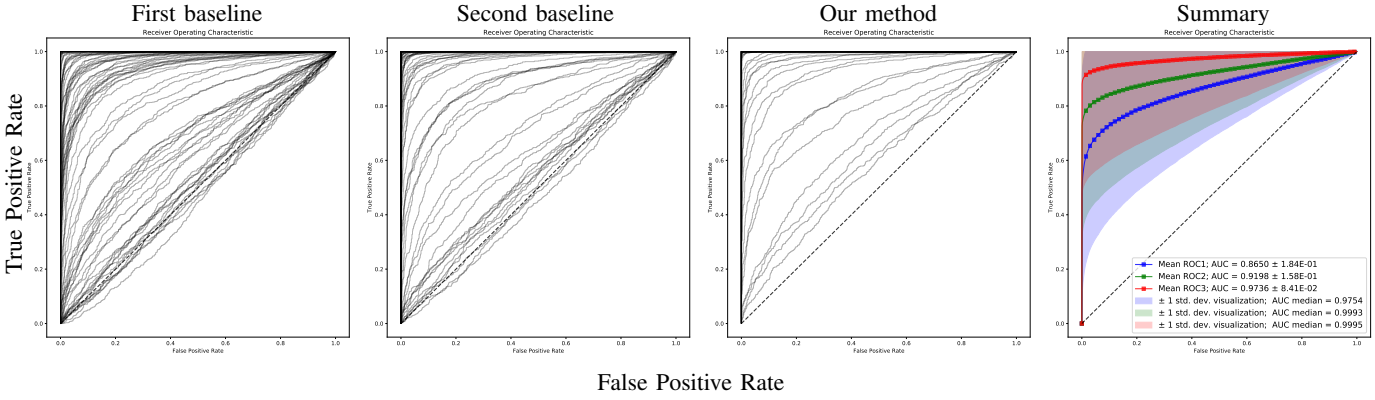

Fig. 3. Same as in Figure 2 except that the tested samples' size is 200.

$Down(3)([1,1,2,2,3,3]) = [1,3]$). $Flat$ is a layer, without learnable parameters, used to reshape the features into a vector so that one can use a fully connected layer afterwards. $Conv$ and $Dense$ are respectively convolutional and fully connected layers and $5DissLayers$ is a layer with 5 Diss-Layers [14].

The autoencoders' layout is: [(15,15),(8,9),45,72,(15,15)]. Output shapes (couples (a,b) or numbers c) are obtained with omitted sampling, flattening and cropping layers. A couple (a,b) indicates a convolutional layer (or the input layer at the beginning) with b channels, each with a features. A unique number c indicates a fully connected layer with c features.

B. Data

In this study, data are of two types: 1) the ICS physical process data and 2) the matrices of reconstruction errors of an 1D-CNN autoencoder that monitors the ICS physical process data. Data of the first type are simulated so that we can gather enough data of the second type to have representative results. Indeed, time series from real-world ICSs would require a large stochastic model—both from the point of view of the autoencoder and the model that authenticates the autoencoder. Yet, since we deal with stochastic models, we need several trained instances to perform a thorough analysis of the performance of the model that authenticates the autoencoder.

Data of types 1 and 2 are created as follows:

1) Simulated ICS physical process data are a binary Multivariate Markov Chain with 15 chains (for 15 sensors/actuators values) that operates a cycle in about 50 time steps in average and comprises 4000 cycles for the training set and 1000 for the test set. This requires much smaller stochastic models than real-world ICSs that typically have tens of sensors/actuators and cycles with hundreds of time steps. Moreover, real-world ICSs also have continuous sensors/actuators data, but for the sake of simplicity we simulate a binary Multivariate Markov Chain where the ICS executes the same sequence of actions in each cycle, and we select five moments (cf. Section III-A).

2) As the method's performance depends on both the genuine autoencoder and the spoofing autoencoder, not only we have to repeat the procedure for several genuine autoencoders, but each genuine autoencoder must be faced with attacks from several spoofing autoencoders. Ten 1D-CNN autoencoders are trained. Then, they alternately play the role of the (single) genuine autoencoder and the 9 spoofing autoencoders, resulting in 90 spoofing attacks which is enough to aggregate representative results. For each autoencoder, a multipath NN classifier and a traditional NN classifier are trained. Finally, samples derived from matrices of reconstruction errors are tested against a sample derived from matrices of errors from the genuine autoencoder only. More precisely, it is a summary

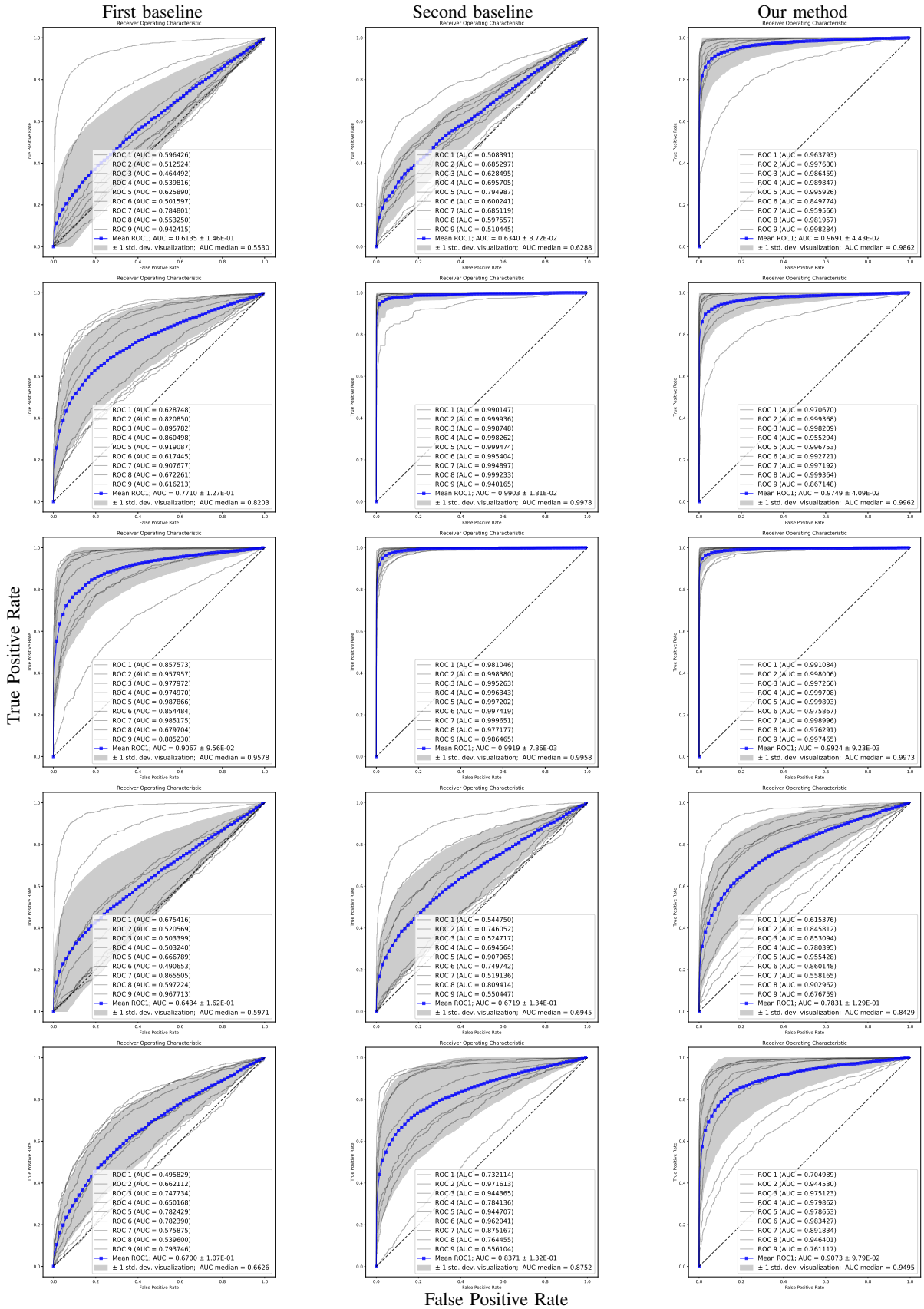

Fig. 4. ROC curves from 1 - p-values of the WMW test on the summary variable of the three methods (one per column) for samples of size 25 tested against a reference sample of size 1000 (see Figure 2 for the summary of the ROC curves per methods). Each row corresponds to a genuine autoencoder, each graph contains 9 ROC curves, one for each spoofing autoencoder.

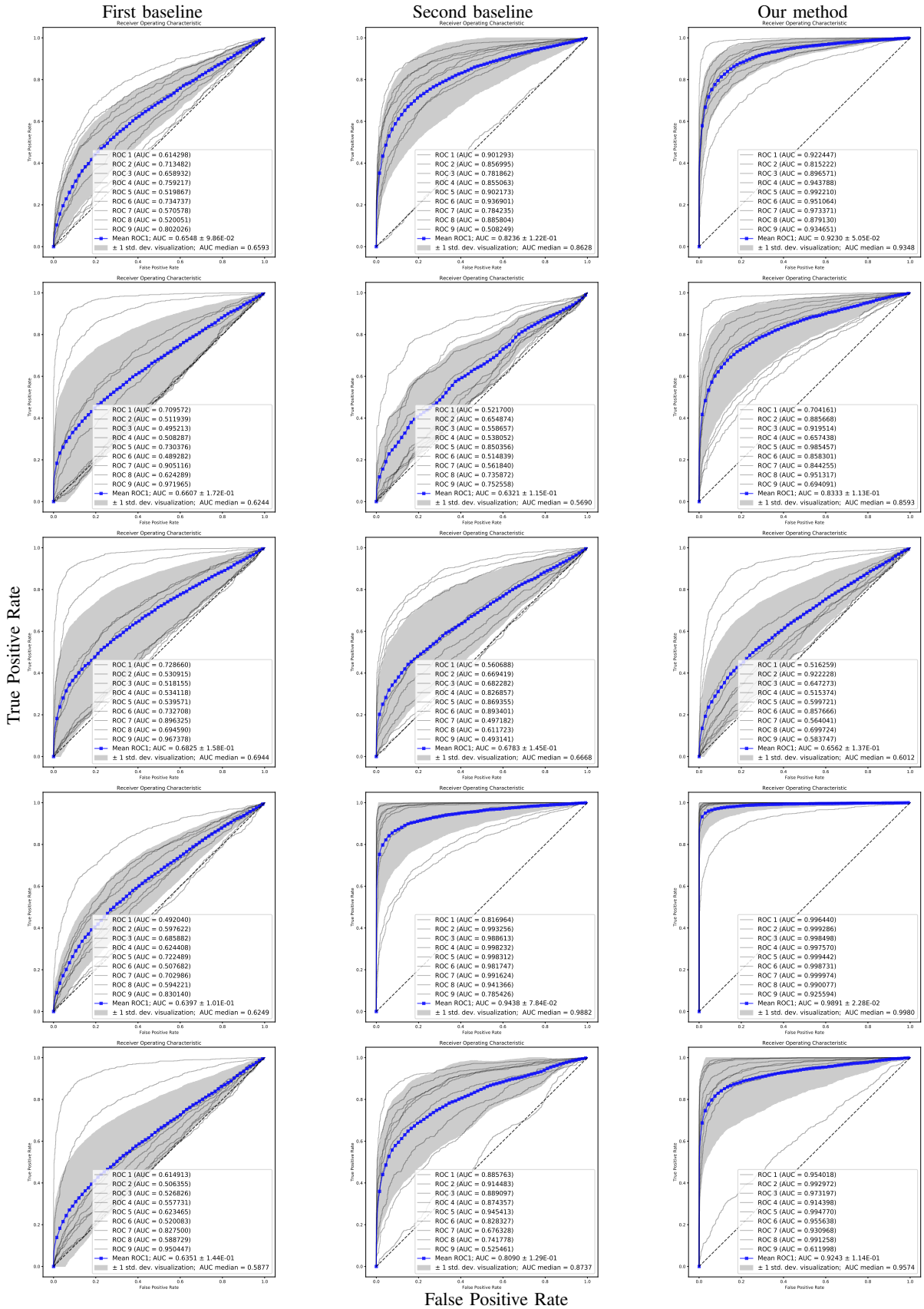

Fig. 5. ROC curves from 1 - p-values of the WMW test test on the summary variable of the three methods (one per column) for samples of size 25 tested against a reference sample of size 1000 (see Figure 2 for the summary of the ROC curves per methods). Each row corresponds to a genuine autoencoder, each graph contains 9 ROC curves, one for each spoofing autoencoder.

of the matrix of errors that is the variable of the WMW test, the confidence measure in the case of the multipath NN classifier and in the case of the second baseline, as well as the mean squared error in the case of the first baseline. Matrices of errors summarized into single values that compose a tested sample are all, either from the genuine autoencoder, either from a unique spoofing autoencoder. This way, a ROC curve can be plotted to visualize how well a genuine autoencoder is distinguished from a particular spoofing autoencoder, but of course this does not change the end result which is the average performance of the model that authenticates the autoencoder.

C. Results

To see which summary variable best distinguishes an abnormal sample (from a spoofing autoencoder) from a normal one (from the genuine autoencoder) with the WMW test, we plot ROC curves from 1 minus the p-value. Each point of a curve comes from a particular anomaly threshold $1 - \alpha$, with α the corresponding FPR (cf. Section II-D). Indeed, 1 minus the p-value can be seen as an anomaly scoring function whose input is a tested sample since, the more abnormal the sample, the lower the p-value due to the consistency of the test (cf. Section II-D). The more the ROC curve is near the top left corner of the graph—or similarly the bigger the Area Under the Curve (AUC)—, the better.

Results of the 90 ROC curves (see Section IV-B) for each the three methods with tested samples of size 25 are shown in Figure 2: 1st baseline's AUC = 0.6877 ± 0.16; 2nd baseline's AUC = 0.8012 ± 0.17; our method's AUC = 0.8953 ± 0.14. Similar graphs for tested samples of size 200 are shown in Figure 3: 1st baseline's AUC = 0.8650 ± 0.18; 2nd baseline's AUC = 0.9198 ± 0.16; our method's AUC = 0.9736 ± 0.08.

D. Discussion

The multipath NN's confidence measure allows the WMW test to detect a spoofing attack with less examples. When used online, it thus detects attacks faster than the baselines.

Moreover, when a ROC curve is too close to the first bisector for tests with 25 examples (Figure 2), the detection of the corresponding spoofing autoencoder is difficult, if not impossible, with more examples (Figure 3). Yet, each ROC curve of the graph related to our method move to the top left corner with the increased size of the tested samples (from Figure 2 to Figure 3), while some ROC curves from the baselines are stuck on the first bisector. This means that some spoofing autoencoder will never be detected by the baselines. In contrast, since the WMW is consistent under a reasonable assumption already mentioned (cf. Section II-D), our method will eventually detect every spoofing autoencoder, even if it takes a large sample and thus a long time for a few of them.

Figures 4 and 5 show the details of Figure 2, that are, for each of the 10 autoencoders, the ROC curves corresponding to the other 9 trying to spoof them. They show that 1) in each case, our method is better than the first baseline and that 2) in only two cases out of the ten cases, the second baseline outperforms our method but only by a small margin.

Finally, it is worth noting that, in practice, when the test on the multipath NN's confidence measure rejects the null hypothesis—meaning that the distribution of the tested sample is likely to be different from the reference sample's distribution—it only indicates there is an anomaly occurring in the matrices of errors of the autoencoder, but it does not tell where this anomaly comes from. It can be from the physical process data or from a spoofing autoencoder. One can only be sure that the first case is wrong when the first baseline does not detect any change in distribution while our method detects a significant change in distribution.

V. Conclusion

In this paper, we propose a method to authenticate the autoencoder of an IDS monitoring an ICS by the use of input-output probes (Section III). This allows to add a layer of security (authentication) without compromising the previous one (anomaly detection). Our method can detect any autoencoder spoofing attacks (Section IV-D) with an arbitrary FPR, yet with a trade-off between the FPR and the time of detection.

Authors contributions

Following the taxonomy from [1]:

Raphaël M.J.I Larsen's contributions: 1) Conceptualization (raised the article's problem: autencoder authentication, conceived the idea of using of the multipath Neural Network to treat this problem) 2) Methodology 3) Investigation 4) Formal Analysis 5) Software 6) Visualization 7) Validation 8) Writing (Original Draft & Editing).

Marc-Oliver Pahl's contributions: 1) Validation 2) Supervision (gave precise guidelines for the first draft presentation that is best for the topic treated) 3) Writing (Review & Editing).

Gouenou Coatrieux's contributions: 1) Conceptualization (conceived the idea of applying some digital forensics principles for authentication) 2) Supervision (helped defining the notion of traceability, which is a notion that led to the idea of autoencoder authentication) 3) Funding acquisition.

References

[1] A. Brand, L. Allen, M. Altman, M. Hlava and J. Scott *Beyond authorship: attribution, contribution, collaboration, and credit* (2015) Learned Publishing, 28(2), 151-155.

[2] C.S. Wickramasinghe, D.L. Marino, K. Amarasinghe and M. Manic, *Generalization of deep learning for cyber-physical system security: A survey* IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society, IEEE, 2018.

[3] C. Wressnegger, A. Kellner and K. Rieck, *ZOE: Content-based Anomaly Detection for Industrial Control Systems* 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, 2018.

[4] D.S. Berman, A.L. Buczak, J.S. Chavis and C.L. Corbett, *A survey of deep learning methods for cyber security*, (Vol. 10). Information, Multidisciplinary Digital Publishing Institute, 2019.

[5] D. Denning and P.G. Neumann, *Requirements and model for IDES-a real-time intrusion-detection expert system* SRI International Menlo Park, 1985.

[6] F. Wilcoxon, *Individual Comparisons by Ranking Methods*, (Vol. 1). Biometrics, 1945.

[7] G. Alain and Y. Bengio, *Understanding intermediate layers using linear classifier probes* https://openreview.net/forum?id=HJ4-rAVtl, ICLR, 2017.

[8] H.B. Mann and D.R. Whitney, *On a test of whether one of two random variables is stochastically larger than the other* The annals of mathematical statistics, JSTOR, 1947.

[9] I. Corona, G. Giacinto, and F. Roli,, *Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues*, (Vol. 239). Information Sciences, Elsevier, 2013.

[10] J. Gardiner and S. Nagaraja, *On the security of machine learning in malware C&C detection: A survey*, (Vol. 49). ACM Computing Surveys (CSUR), ACM New York, NY, USA, 2016.

[11] K. Andrej. Convolutional neural networks (cnns/convnets), https://cs231n.github.io/convolutional-networks/, 2016.

[12] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández and E. Vázquez, *Anomaly-based network intrusion detection: Techniques, systems and challenges*, (Vol. 28). Computer & Security, Elsevier, 2009.

[13] R. Mitchell and I. Chen, *A survey of intrusion detection techniques for cyber-physical systems* (Vol. 46). ACM Computing Surveys (CSUR), ACM New York, NY, USA, 2014.

[14] R.M.J.I. Larsen, *Hyper-Neurons and Confidence Through Path Validation* 2021, https://gitlab.imt-atlantique.fr/chaire-cyber-cni-public/host4paper/-/raw/c501b6940030b48c5589f5380930b4f26e77d256/papers/Hyper-NeuronsandCTPV.pdf

[15] S. Axelsson, *Research in intrusion-detection systems: A survey*, (Vol. 120). Technical report 98–17. Department of Computer Engineering, Chalmers University of Technology, 1998.

[16] T.H. Ptacek and T.N. Newsham, *Insertion, evasion, and denial of service: Eluding network intrusion detection* Secure Networks inc Calgary Alberta, 1998.

[17] Y. Hu, A. Yang, and H. Li, Y. Sunand L. Sun, *A survey of intrusion detection on industrial control systems* (Vol. 14). International Journal of Distributed Sensor Networks, SAGE Publications Sage UK: London, England, 2018.

[18] Y. Luo, Y. Xiao, L. Cheng, G. Peng, D.D. Yao, *Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities* arXiv preprint arXiv:2003.13213, 2020.