Anomaly Detection and Intrusion Detection Systems in a simulated ICS environment

Nicola Scremin, Simone Zerbini
University of Padua, Italy

***Abstract*—Safety for an Industrial Control System is fundamental and anomaly detection is an approach extensively researched for this purpose. The goal of this paper is to implement and evaluate a real time intrusion detection system, which exploits anomaly detection in time-series and unsupervised learning. To generate our datasets, we used DHALSIM, a new, advanced and promising simulator that could represent a large area for future experiments. In fact, it presents a functionality which permit to generate a Dataset under Man In the Middle (MITM) attack.**

Keywords: Anomaly Detection, Intrusion detection, IDS, Industrial Control System, ICS, MITM.

I. Introduction

The increase in interconnection and automation in Industrial Control Systems (ICSs) has paved the way for new cyber-physical threat scenarios. Furthermore, specific ICSs categorized as critical infrastructures, like refineries, power plants, nuclear plants, and water distribution systems, can cause serious problems to the population and the environment if damaged.

We implemented an Intrusion Detection System (IDS) analyzing the network traffic of a simulated ICS with an unsupervised time-series anomaly detection technique. The motivations of our choices are illustrated below.

Anomaly detection refers to the task of discovering anomalies in the data. An anomaly is an observation which deviates from the distribution of the data set. Regarding cyber-security, anomaly detection is used for intrusion detection, since an anomaly could be due to attack attempts by malicious parties. The main advantage of using anomaly detection for intrusion detection is its capability of detect also unknown attacks since it does not aim to recognize a specific attack, but it only looks for a departure from the normal operating conditions of the system. However, the anomaly detection approach has its drawbacks such as the system complexity, high false alarm rate and the difficulty of detecting which event triggers those alarms. In a cyber-physical system we can apply anomaly detection both on the physical data and network traffic.

We have chosen to use a simulator to generate our datasets with the aim of operating in a more controlled and reproducible environment, in order to avoid the problems highlighted in the paper [1], i.e., it is hard to obtain, from a real testbed, two datasets in which the physical process starts from the same point and evolves in the same way. We used DHALSIM, a new and advanced simulator which has not yet been widely used for analyzes of this type, and therefore it can represent a wide area for future experiments.

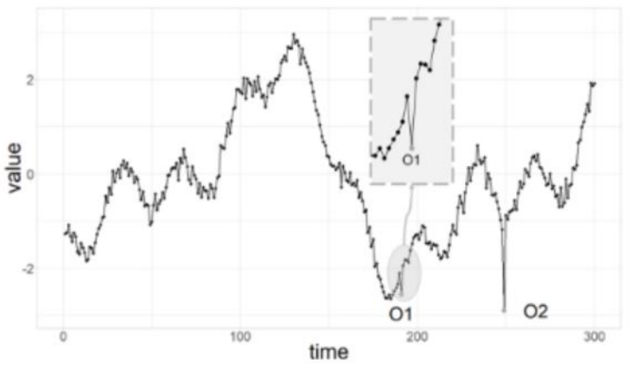

Figure 1: Typical example of outliers in a time-series

Machine learning is widely used for anomaly detection, using both supervised and unsupervised techniques. Theoretically, supervised techniques perform better in terms of detection and false detection rate, but they have some technical issues. In fact, supervised techniques need a large labelled training set containing data that cover all the possible areas, which makes them ineffective against new threats. Moreover, it is challenging to obtain accurate labels, especially in our scenario where it is difficult establish a priori if a transmission is related to an attack or not [2]. On the other hand, unsupervised techniques aim to learn the normal behaviour of the data in order to discriminate the anomalies, in this way they can detect both known and unknown anomalies. Nevertheless, supervised learning is the most used approach in this field [3], that is why we chose an unsupervised learning.

The data we are going to analyze for this project are time-series, that is a ordered sequence of observation, in our case a sequence of network packets. Unlike in a static anomaly detection, in which an anomaly can be seen simply as a point that differs a lot from the others, in a time series an anomaly is a point that does not follow the trend of the series (not necessarily different from the other points). In Fig. 1 is represented an example of outliers in a time series.

One way of doing anomaly detection with time series data is by building a predictive model using the historical data to estimate the trend and predict the future points. By doing so we are able to determine if a point is anomalous by quantifying how far it differs from our prediction. The effectiveness of a time series model in this context has already been demonstrated [4], in any case we compared our results with some non-time-series methods.

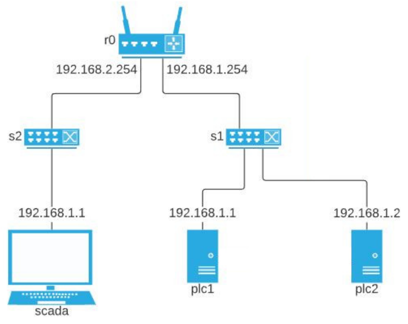

Figure 2: Topology of the simulated network

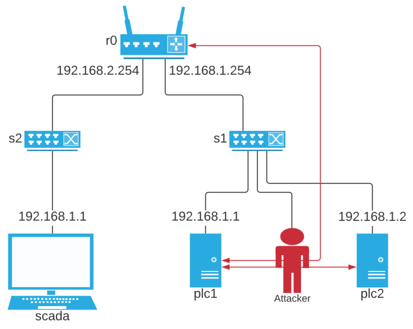

Figure 3: Topology with an attacker.

The rest of this paper is organized as follows. Section II describes the simulator and our configuration. Section III illustrates how we proceeded to carry out the analysis. Section IV gives a critical discussion on our results. Section VI concludes the paper.

II. Simulator

The project consists in developing an Anomaly Detection and Intrusion Detection System in a cyber-physical system environment, more precisely in an Industrial Control System (ICS) which controls a water distribution system. For this purpose, we used the DHALSIM simulator [5] to generate both the training and the test set.

DHALSIM simulates physical, control, and network processes. In order to do so it interfaces an accurate hydraulic simulator (WNTR) with a state-of-the-art industrial network emulator (MiniCPS). Moreover, it provides a platform for simulating attacks that affect both physical and network processes. In particular, we simulate man-in-the-middle attacks, i.e. attacks in which the attacker modifies the responses of a PLC, altering the value of a sensor or actuator. The simulator offers two different implementations of mitm attacks, unfortunately their description is not very clear. They are described as follows [6]:

- Man-in-the-middle (MITM) attacks are attacks where the attacker will sit in between a PLC and its connected switch. The attacker will then route host a CPPPO server and respond to the CIP requests for the PLC.
- Naive Man-in-the-middle (naive MITM) attacks are attacks where the attacker will sit in between a PLC and its connected switch. The attacker will then route all TCP packets that are destined for the PLC through itself and can for example modify the responses to the other PLCs.

A. Our topology

We have chosen a simple network topology with only 2 PLCs (Fig. 2), this choice is mainly due to the attempt to reduce the computational burden of the simulation. The first PLC controls a sensor which measures the water level of a tank, while the second PLC controls two pumps. For the test set, we have implemented 4 man-in-the-middle attacks with different timings and targets, reported in table I.

Index	Target	Action	Implementation
1	PLC2	turn off pump 1	mitm
2	PLC1	set the sensor value to 0.1	naive mitm
3	PLC2	turn on pump 1 and 2	naive mitm
4	PLC1	set the sensor value to 6.4	mitm

Table I: Description of the attacks.

III. Methodology

In this section are illustrated the steps used to carry out the analysis.

A. Data collection

For the training set, we ran the simulation for 10000 iterations, resulting in one hour and 20 minutes of network traffic. On the other hand, for the test set, we ran the simulation, with the 4 attacks, for 7000 iterations, obtaining one hour and 40 minutes of network traffic. An iteration corresponds to one second of physical simulation, while the packets' timestamp corresponds to the real time in which they are generated. One important thing to highlight is that the time to complete x iterations is proportional to the computational power of the machine in which the simulation is running. Moreover, it can be seen that the simulation of the test set took longer even though it consists of fewer iterations. This is because the implementation of the attacks requires adding some nodes in the network, obtaining a more complex topology and so it requires more computational power. This problem and its possible consequences are analyzed in more detail in section IV.

B. Features extraction

We have carried out our analysis on the traffic recorded by the PLCs. The analysis of network traffic on a dataset containing every single packet is inefficient and does not perform well, so we applied the CICFlowMeter tool [7] to group the packages into bidirectional flows for which 76 features are extracted. Before doing that, we filtered the packets by removing those that use IPv6 protocol.

C. Features selection

With the aim of obtaining a simpler computation, we have chosen only 6 features among the 76 extracted by CICFlowMeter. For the non-time series correlated methods, we applied some features selection techniques as PCA and selectKbest. Unfortunately, these techniques cannot be applied for a method which works in the time series domain, so, for our time series method, we chosen the features mainly by a trial and error approach, adopting as starting point the features used in the paper [8]. We have also observed that our dataset contains a lot of constant or strongly correlated features, and so useless. The chosen features are: flow duration, total size of packet in forward direction, mean size of packet in forward direction, mean size of packet in backward direction, number of backward packets per second, and the total number of bytes sent in initial window in the backward direction.

D. Detection

We adopted a Vector Autoregression model (VAR), in which each feature's value is modelled as a linear combination of past values of itself and the past values of the other features. The prediction of the observation at time t is given by the following equation:

$$y_t = c + A_1 y_{t-1} + ... + A_p y_{t-p} + e_t \tag{1}$$

Where c is a constant vector serving as intercept, A_i are time-invariant matrices, e is a vector of error terms, and p is the lag of the model, that is , how far the model goes into the past.

In order to compare the results with some non-time-series correlated methods, we have tried One Class SVM (OCSVM), Local Outlier Factor (LOF) and Isolation Forest (IF).

Isolation Forests are build based on decision trees, following the theory that anomalies are "few and different" data points. The main idea, which is different from other popular outlier detection methods, is that Isolation Forest explicitly identifies anomalies instead of profiling normal data points. Going deeper in details, IF is nothing more than an ensemble of binary trees (each called $iTree$, Isolation Tree). The iTree consist of the following steps:

1) Random sub-sample of the data is selected and assigned to binary tree.
2) Select random feature and to breach on a random threshold (any value between minimum and maximum is allowed).
3) If data point is more than the threshold, it goes on the right otherwise on the left branch.
4) This process from step 2 is recursively done until each data point is completely isolated or max depth is reached.
5) Steps 1-4 are repeated to build random binary trees

After that, an ensemble of iTrees is created and the model training is finally complete. The samples which end up in short branches indicate anomalies as it was easier for the tree to separate them from other observation. Instead, if sample is placed in a deeper position means it is a benign one.

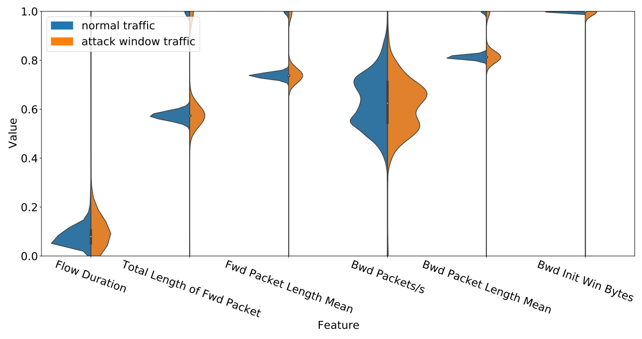

Figure 4: Violinplot of the chosen features. Normal flows in blue, traffic from the attacks windows in orange. The small differences that we can notice are due to the imbalance of the classes, i.e., the normal traffic distributions have a better approximation.

Local Outlier Factor (LOF), instead, is a metric that represents how likely a certain data point is anomaly/outlier. LOF takes care about neighbors and in particular it requires a parameter k which is the number of neighbors considered for the computation. Now we have to define $k - distance$ which is the distance of a point to the k-th closest point. $k - distance$ is used to calculate the $reachability\ distance$ which simply measure the maximum between the distance of two points and $k - distance$. It just introduce a "smoothing factor". After that, $reachability\ distance$ is used to define the $localreachabilitydensity$, lrd. In particular we first calculate the $reachability\ distance$ of one point to all its k neighbors and take the average of it and, lrd, is simply the inverse. Technically, lrd tells how far our point is to reach the next point or a cluster of points. The lower it is, the less dense it is and longer we have to travel. Now we can finally define LOF: the lrd of each point is then compared to the lrd of their k neighbors. It is basically the average ration of the lrds of the neighbors of a point to the lrd of that specific point. So, the LOF of a point tells the density of this point compared to the density of its neighbors. If the density of a point is much smaller than the densities of its neighbors (LOF >> 1), the point is far from dense areas and, hence, an outlier.

SVMs use hyperplanes in multi-dimensional space to separate one class of observations from another. More recently SVM is started to be used in one class problems (OCSVM), where all data belong to a single class. In this case the algorithm does not learn how to classify the data but learn what is "normal". Doing so, when we present a new sample to the algorithm it can identify whether it is benign or not. If not, the new data is labeled as outlier.

Finally, we used the timestamps reported in the files "ground_truth" and "scada_values" to bind iteration and time.

IV. Results

With the non-time series correlated methods we couldn't obtain any proper results. This is due to the fact that, as shown in Fig. 4, the features of the attack-related flows follow the same distribution as those of normal flows. Therefore, it is not enough to look for outlier flows, but we have to look for anomalies in the trend of the flows' features.

Let's now talk about the results of the VAR model. In Fig. 5 we plotted the squared error of the model's predictions with respect to the test set traffic. Then, we proceeded by adopting 2 different approaches for choosing the threshold. First we tried with an adaptive threshold calculated as the sum of the mean and the standard deviation of the squared errors, the obtained result is shown in Fig 6. It can be notice that the model detect at least an anomaly in each attack. In particular, in the first and fourth attack, the model is able to detect anomalies for the entire duration of the attacks. Probably, this is due to the fact that these attacks have a different implementation than the others. Unfortunately, it can also be noted the large presence of false positives. Then we tried a static threshold, choosing the highest squared error obtained analyzing a traffic without attacks. The results are reported in Fig. 7, it can be notice that we have less false positive, but this approach is feasible only in an ICS scenario where the traffic is mostly constant.

Trying to figure out what was causing all those false positives, we have tried to remove from the pcap files the ARP packets used to carry out the attacks (the simulator implements MITM attacks via ARP spoofing). As result (Fig. 8), we obtained a model which does not produce false positives while detecting at least an anomaly in the first 3 attacks. This is not an improvement nor a solution to our problem, but we can infer that the false positives are a consequences of the attacks. Thus, there is the possibility that our model does not produce any false positive in a scenario without attacks. It is clear that an attack produces a sort of perturbation in the model that causes the detection of false positives, unfortunately we only have less than 2h of traffic and therefore we are not able to establish how long these perturbations propagate over time.

Another possible source of error is due to the fact that the simulation of the test set takes longer than that of the training set (as already mentioned in section III-A). This means that the time intervals between physical events result dilated, and therefore also between the traffic that follows. This "temporal dilation" can be considered by the model an anomaly in itself, and so it is a possible misclassification factor.

We also evaluated the models in terms of F1 score and accuracy, reported in table II, but we believe this is not a good way to evaluate these techniques. In fact, the labeling is very approximate and labels all flows within the attacks' windows as attack, most likely including flows not related to it, so we prefer a qualitative evaluation. About this, we can say that our model detects attacks by signaling a positive at the beginning of the attack's window, and thus even a quick and stealth attack can be detected.

V. Future works

In future related works we could start from these project to improve the results. We are going to give some improvements which are good from our point of view:

- Modify the simulator in such a way that as timestamp of the generated packets it uses a time that reflects that of

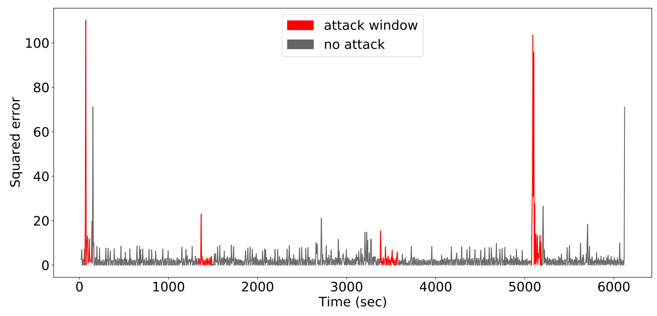

Figure 5: VAR model's output. It can be noticed that the two naive-mitm attacks are much more stealthy than the others.

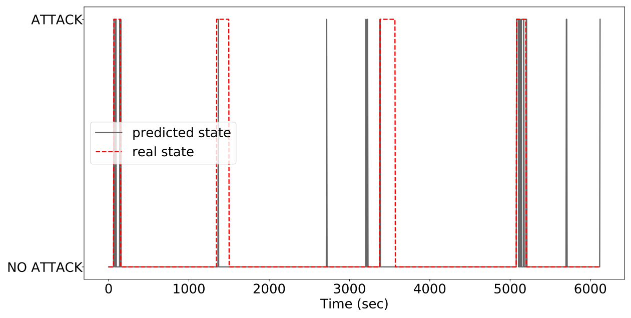

Figure 7: Our model prediction using a fixed threshold.

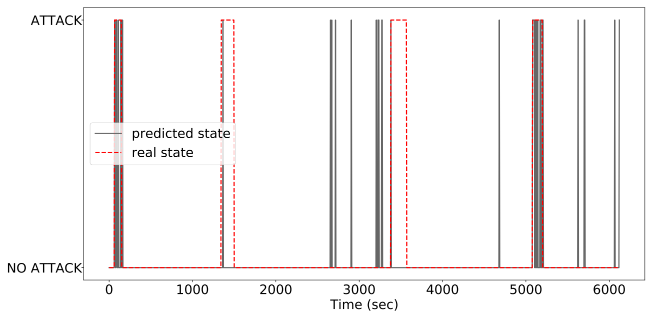

Figure 6: Our model prediction using an adaptive threshold.

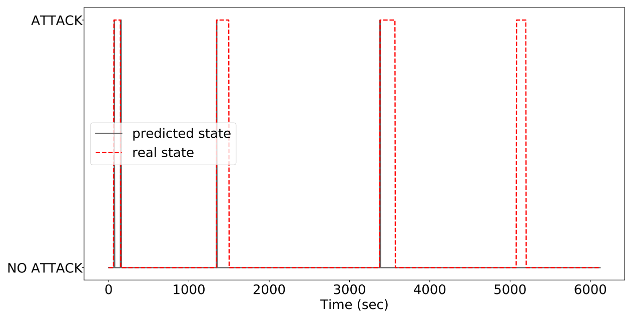

Figure 8: Result obtained removing the ARP packets.

Methods	Accuracy	F1-score
One Class SVM	0.446	0.081
Isolation Forest	0.958	0.002
Local Outlier Factor	0.952	0.019
VAR adaptive threshold	0.909	0.245
VAR fixed threshold	0.915	0.224

Table II: Accuracy and F1-score for each method. It can be seen that the accuracy of OCSVM is close to 50%, i.e., it is like a random guess. Instead, LOF and IF have good accuracy but this does not mean that they work well: basically these accuracies are obtained by classifying all flows as non-attack, then the unbalancing of the classes does the trick.

the physical simulation, so as to make the timestamps independent from the computational power of the machine.
- Trying to implement some other attacks, like MITM mDNS Spoofing.

VI. Conclusions

By summing up in this paper we briefly explained what IDS is and on which machine learning concepts is based on. After that we exploited DHALSIM that is a tool to generate a traffic data in a real cyber-physical scenario. One of the most important functionalities of this program is that it also implement MITM attack. In this project in fact we exploited these attacks in order to build a real time intrusion detection system. From the results we saw easily that time-series methods (VAR in particular) work really better than non-time series detection methods as LOF, OCSVM and Isolation Forest. Since in an IDS system is quite difficult to achieve perfect performance which means to only identify correctly the attacks without false positive, we are enough satisfied by our results. If we want to be meticulous our model initially noted too many False Positive (which in a real scenario means to waste time for checking if an attack is runs, when it is not actually). Playing with some thresholds we further improved our results. Moreover it is also better to have FP than FN. False Negative means that an attack is undetected and, in our research, never happen.

References

[1] F. Turrin, A. Erba, N. O. Tippenhauer, and M. Conti, "A statistical analysis framework for ics process datasets," *In Proceedings of the 2020 Joint Workshop on CPS&IoT Security and Privacy (CPSIOTSEC'20). Association for Computing Machinery, New York*, pp. 25–30, 2020.

[2] G. Bernieri, M. Conti, and F. Turrin, "Evaluation of machine learning algorithms for anomaly detection in industrial networks," *2019 IEEE International Symposium on Measurements & Networking (M&N)*, pp. 1–6, 2019.

[3] O. A. Alimi, K. Ouahada, A. M. Abu-Mahfouz, S. Rimer, and K. O. A. Alimi, "A review of research works on supervised learning algorithms for scada intrusion detection and classification," *Sustainability*, 2021.

[4] A. Agiollo, M. Conti, P. Kaliyar, T. N. Lin, and L. Pajola, "Detonar: Detection of routing attacks in rpl-based iot," *IEEE Transactions on Network and Service Management*, pp. 1178–1190, 2021.

[5] A. Murillo, R. Taormina, N. Tippenhauer, and S. Galelli, "Co-simulating physical processes and network data for high-fidelity cyber-security experiments," in *Sixth Annual Industrial Control System Security (ICSS) Workshop*, 2020, pp. 13–20.

[6] ——, "Co-simulating physical processes and network data for high-fidelity cyber-security experiments," 2020. [Online]. Available: https://github.com/afmurillo/DHALSIM

[7] A. Habibi Lashkari, "Cicflowmeter-v4.0 (formerly known as iscxflowmeter) is a network traffic bi-flow generator and analyser for anomaly detection. https://github.com/iscx/cicflowmeter," 2018.

[8] 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," *Association for Computing Machinery*, 2020.