Intrusion Detection in Cyber Physical Systems Based on Process Modelling

Tamás Holczer, András Gazdag and György Miru
Laboratory of Cryptography and System Security, Budapest University of Technology and Economics, Hungary
holczer@crysys.hu
agazdag@crysys.hu
miru@crysys.hu

Abstract: Cyber physical systems (CPS) are used to control chemical processes, and can be found in manufacturing, civil infrastructure, energy industry, transportation and in many more places. There is one common characteristic in these areas, their operation is critical as a malfunction can potential be life-threatening. In the past, an attack against the cyber part of the systems can lead to physical consequences. The first well known attack against a CPS was Stuxnet in 2010. It is challenging to develop countermeasures in this field without endangering the normal operation of the underlying system. In our research, our goal was to detect attacks without interfering with the cyber physical systems in any way. This can be realized by an anomaly detection system using passive network monitoring. Our approach is based on analysing the state of the physical process by interpreting the communication between the control system and the supervisory system. This state can be compared to a model based prediction of the system, which can serve as a solid base for intrusion detection. In order to realize our intrusion detection system, a testbed was built based on widely used Siemens PLCs. Our implementation consists of three main parts. The first task is to understand the network communication in order to gain information about the controlled process. This was realized by analysing and deeply understanding the publicly undocumented Siemens management protocol. The resulting protocol parser was integrated into the widely-used Bro network security monitoring framework. Gathering information about the process state for a prolonged time creates time series. With these time series, as the second step, statistical models of the physical process can be built to predict future states. As the final step, the new states of the physical process can be compared with the predicted states. Significant differences can be considered as an indicator of compromise.

Keywords: intrusion detection, process modelling, cyber physical system, anomaly detection

1. Introduction and background

Industrial control systems (ICS) are special cases of cyber physical systems (CPS). The security of CPSs is an important issue especially if it is an ICS. ICSs can control factories, electric grids or many other important systems. It is common in these systems, that an attack can have tremendous effects.

It is hard to develop defence mechanisms in this field without endangering the normal operation of the underlying system. For this reason, it was decided to implement a passive detection system, which cannot interfere with the normal operation of the controlled process, but can trigger an alarm, in case of suspicious events.

In order to realize our intrusion detection system, a testbed was built based on widely used Siemens PLCs. Our implementation consists of three main parts. The first task is to understand the network communication in order to gain information about the controlled process. This was realized by analysing and deeply understanding the publicly undocumented Siemens management protocol. The resulting protocol parser was integrated into the widely-used Bro network security monitoring framework. Gathering information about the process state for a prolonged time creates time series. With these time series, as the second step, we can build statistical models of the physical process to predict future states. As the final step, the new states of the physical process can be compared with the predicted states. Significant differences can be considered as an indicator of compromise.

The following attacker model was used throughout our work. The attacker has direct access to the physical process. It can be a physical or logical access as well, but it modifies the measured variables of the process. In most cases, it is implemented by modifying the process and masquerading the process output from the operator by modifying the output of the human machine interface (HMI). It was assumed that the real sensor readings could be grabbed from the network communication. This kind of attack can be detected by this approach. In this attacker model, it is assumed that there is a clear learning phase before any attack is done.

In the next sections it is shown how the system used differs from the state of the art solutions, how the system was built, and finally how accurately it could detect attacks.

2. State of the art

In conventional IT intrusion detection has a rich tradition. Axelsson *et al*. (2000) reviews and categorizes the existing IDS solutions based on the used detection principle and certain operational aspects of the system. Debar *et al.* (1999) further refines the taxonomy of intrusion detection systems and defines families according to their properties.

Intrusion detection in industrial control systems is a relatively new field of study, however there are many notable articles on the subject. Zhu *et al.* (2010) investigates the SCADA specific intrusion detection techniques and systems, provides a definition, taxonomy and a set of metrics to compare existing solutions. The paper also tries to ease interoperability between traditional IT security and ICS research by providing cross discipline insight to the detection problems of both fields.

Valli (2009) outlines an approach of using traditional IT applications (Snort, nepenthes, honeyd) to create a resilient layered defence application for control system networks. Verba *et al.* (2008) proposes SCADA specific anomaly and signature based packet inspection mechanism and traffic flow analysis with fine protocol granularity. Garitano et al. (2011) reviews research done in the field of anomaly based intrusion detection in ICS applications and evaluates the viability of such methods. Yang *et al.* (2006) describes an anomaly based IDS for control systems, the implemented method uses auto associative kernel regression model with statistical probability ratio test to detect anomalies in the packet flow. The proposed system is applied on a simulated real time SCADA system and the test results are evaluated.

Kiss *et al.* (2014) uses data clustering and Big Data technologies for real-time analysis of control network and sensor data in order to identify physical threats. Hadžiosmanović *et al.* (2014) introduces a system to detect anomalies in the monitored process variables. The proposed method extracts the data from the network and classifies the variables into three different groups. He identifies constant, discrete in a domain and dynamic variables and uses auto regression to detect anomalies.

3. System design

Our system consists of three main parts. These main parts are shown on the following figure.

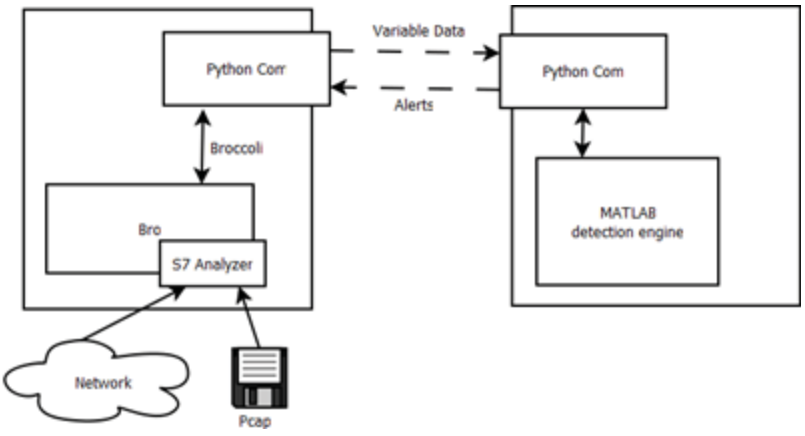

Figure 1: System overview

The first part is responsible for intercepting the network traffic and analysing it. This part is realized by a newly developed Bro analyser. The second part is responsible for the communication between the Bro analyser and the detection engine. This part is implemented by some Python scripts. The third part is responsible for the anomaly detection. This part is implemented in Matlab. In the following sections, these parts are introduced, before evaluating the whole system.

3.1 Message interception

As discussed in the previous sections, the general design of the proposed IDS requires the process data to be extracted from the network communication. As a result, the proposed system needs to be able to observe and understand the communication between the field devices and the entities managing them. Capturing the traffic of large, distributed IP networks is a nontrivial task. However there are multiple existing solutions and papers covering the subject. In the following discussion we assume that the entire network traffic is available in a capture file or monitored to a single interface of a dedicated PC.

The IDS aims to protect ICS systems built with specific Siemens equipment that uses the proprietary S7 protocol. This is a closed, undocumented protocol. The reverse engineering of the protocol was required to an extent to be able to extract valuable process related information from the communication. The main technical challenges we faced were to get an understanding of the S7 protocol and create a software piece that can parse it. Also, the software needed to read variable data that can be passed to the detection engine. The rest of this segment shortly describes the reverse engineering process and details the development of the protocol parser.

The Siemens S7 protocol is widely used in the industry and it has been the interest of many to acquire a deep knowledge of the protocol. Although, at the time of this writing no comprehensive documentation exists there are notable projects that need to be mentioned. Davide Nardella has created an open source communication library the Snap7 (Nardella, 2015), which implements basic communication scenarios. The library comes with the extensive documentation of the basic structure of the S7 protocol. Another project is the S7 Wireshark dissector by Thomas W. which covers most of the protocol and its source code and contains a lengthy list of protocol constants. None of these projects are complete, yet they are invaluable when dealing with S7 communication.

The Siemens TIA portal and WinCC Advanced were used as communication masters and an S-300 series PLC as the slave device, and observed the communication between them in Wireshark network analyzer. The main challenge we encountered was that the S7 protocol turned out to be bloated and redundant in functionality. The protocol contains multiple addressing modes and multiple ways to execute the same actions. We had to uncover each of these in order to not miss any valuable data exchange between the entities.

The software analyzing the protocol is required to process live network streams or capture files, extract the needed variable data and send it the Matlab detection engine. Also, there are secondary requirements such as logging raw data and detection events and presenting alerts in case an intrusion is suspected. To implement these functionalities multiple approaches have been considered, these are the following:

- Create a custom program from scratch
- Use an existing scriptable network analyzer
- Use an existing and extensible IDS implementation

All these methods have their pros and cons, a custom program provides the most flexibility, however it might require considerably more effort to implement, integrate and test. Building the protocol analyzer on top of an existing IDS provides the benefit of having a tested and established code base fulfilling all the secondary requirements, thus greatly reducing the amount of the implementation work. It was decided that the additional benefit of an existing IDS outshines the flexibility provided by the custom implementation.

Multiple major IDS programs were considered when deciding on the base software. The Bro IDS for the S7 protocol parsing was chosen because it is a well-established, versatile, scriptable detection framework. In addition, it provides means through communication APIs to pass the extracted data to further processing which was a crucial requirement for the detector. It also appeared to be more flexible and customizable than the other candidates.

The Bro inner structure follows the conventional actor model, where each actor can publish or subscribe to certain events. Bro provides a domain specific scripting language that is used to define event handlers or publish new events, create data structures, write arbitrary program code and use the services of existing frameworks. These frameworks provide a variety of functionality such as logging, creating alarms and notices, geolocating IP addresses and offer signature based detection.

The performance critical parts of the Bro IDS are written in C++, and there is an interoperability layer between the native and the scripted stack. The Bro framework is extensible with plugins which can contain native code and bro scripts as well, they can implement arbitrary functionality. One use of such plugins is to write protocol analyzers for communication protocols not yet implemented in Bro. Inside the program, these analyzers are used to parse the different network protocols, extract the required information and raise events that are later handled in the scripted layer. Unfortunately, this part of the bro development is not yet documented; however there are plenty of different analyzer examples in the Bro sources and plugins can access the same APIs and classes as any source code.

We have decided to implement the protocol parser in such plugin. The analyzer receives the S7 packets and the connection state information, which allows it to store data between calls and map specific requests to their replies. This is essential when extracting the process variables, due to the way memory reading is implemented in the S7 protocol. During these memory reads only the request sent by the master contains the variable address and size information the slave replies with the raw data bytes. The variable data extracted by the analyzer is sent to the upper layers for further processing. The plugin also parses other, non-process-variable-related S7 communication events, which can be used for conventional rule based intrusion detection. Such events are: program block modification, diagnostic data read, PLC control commands, firmware update, read/write request to certain memory areas, password authentication.

The data and connection events are processed by upper layer bro scripts and logged by the logging framework. The Bro IDS is equipped with the Bro Communication Client Library (BroCCoLi), the library can be used to send to or receive events from other programs. It is written in C, however it has python and ruby bindings. The S7 data handler bro script sends the extracted data to the python script which calls the Matlab detection engine and sends back detection events if there is any. The data handler script can raise alerts and notices based on these detection events. The event flow inside and outside Bro is presented in the picture below.

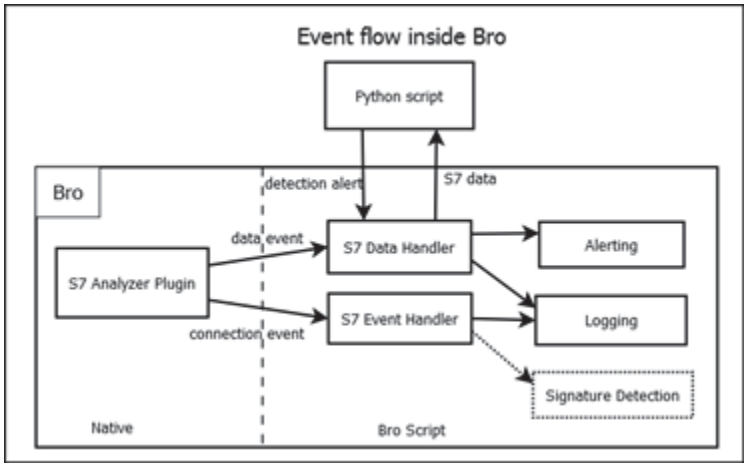

Figure 2: Event flow inside bro

3.2 Middleware

The task to solve for the middleware was to establish communication between the Bro part of the system and the Matlab engine. This problem had to be solved within the boundaries of various further prerequisite.

The Bro engine runs on Linux, whereas our Matlab version runs on Windows and has a number of interfaces for communication. We have chosen the python language to build the bridge between the systems.

On the Bro side an inter process communication had to be used to establish a connection. Fortunately, Bro comes with support for that called the BroCCoLi library. This library lets the developer call an external python function to export Bro data. It works in the other way as well, allowing initiating Bro events from the outside. The remaining task to be solved was to convert the internal data types and structures from Bro to standard python types. This was required to be able to send the data through the network connection.

On the Matlab part another inter process communication realization was required. Matlab has a plugin to work with python scripts. This wraps the Matlab engine into a python object allowing the user to execute Matlab functions from within the python context. A further task to solve was to interpret the data for the Matlab engine and then evaluate the result.

Based on the Matlab engine decision a returning network communication may be required to alert the Bro engine, and with that the network operator as well, about a potential attack.

3.3 Model based detection

In intrusion detection systems, many approaches exist to detect malicious activities. The two main approaches are the signature based detection and the anomaly based detection. Signature based detection uses previously known intrusion signatures, while anomaly detection detects the changed behaviour of the system. In this work we used anomaly detection based approach, because the attack signatures are not previously known in industrial control systems.

Many anomaly detection based approaches exist in the literature, but the model based approach suits best our purposes. Its main idea is to build a model of the investigated process, and detect if the model and the real measurements are diverging. If the difference between the model and the real measurements is over a threshold, an alarm can be raised. In our case the model is built using clean measurements, when no attack is assumed in the system. The main building blocks are shown on the following figure.

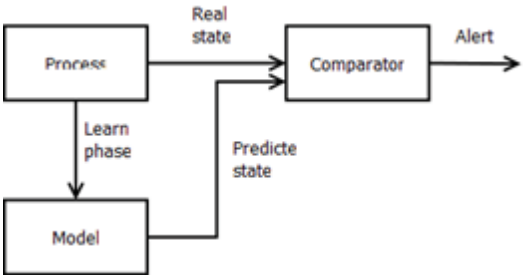

Figure 3: Model based detection

The advantages of model based detection are that it can detect unknown attacks, and the model adapts itself automatically to previously unknown processes. The main disadvantages of model based detection are false alarms and the need for clear traffic.

In the following section, we will show how the model based detection works in industrial control.

4. Evaluation

In this section we will show how our model based intrusion detection systems works in industrial control system environments. First two different approaches are compared. They are the autoregressive–moving-average (ARMA) models and neural networks. The two models are analysed, and the problem of parameter selection is covered as well. After finding the more promising model and parameters, the model is evaluated in a different environment.

4.1 Comparison of models

The first model we used is the autoregressive–moving-average (ARMA) model. It models a weakly stationary stochastic process in terms of two polynomials, one for the auto-regression and the second for the moving average:

$$X_T = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-1} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-1}$$

The other model candidate we used is nonlinear autoregressive neural networks. They can be trained to predict a time series from that series past values.

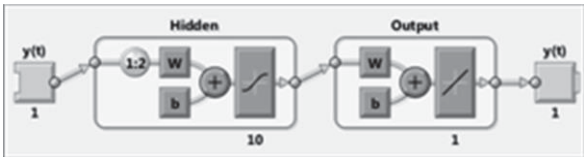

Figure 4: NARNET model

Both models were analysed by a simple process. The process contained a tank, where randomly changing amount of water is filled in. The control process kept the level of water between a minimum and a maximum value by opening and closing a valve. The schematic of the process is shown on the following figure.

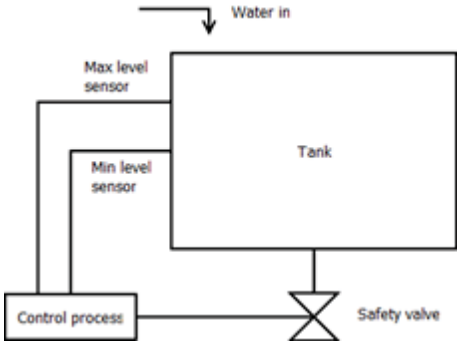

Figure 5: Test process

When deciding between the two models and the parameters, the average accuracy of the model, the standard deviation of the model and required time of running was analysed based on Matlab implementations of the models. The results are shown on the following figures. The ARMA model was analysed with parameters running from 1 to 40 (degree of the polynomials). The parameters of the NARNET model were the number of hidden layers and the delay of the feedback. Every experiment was run hundred times. The value of the fit is the mean value of the hundred runs.

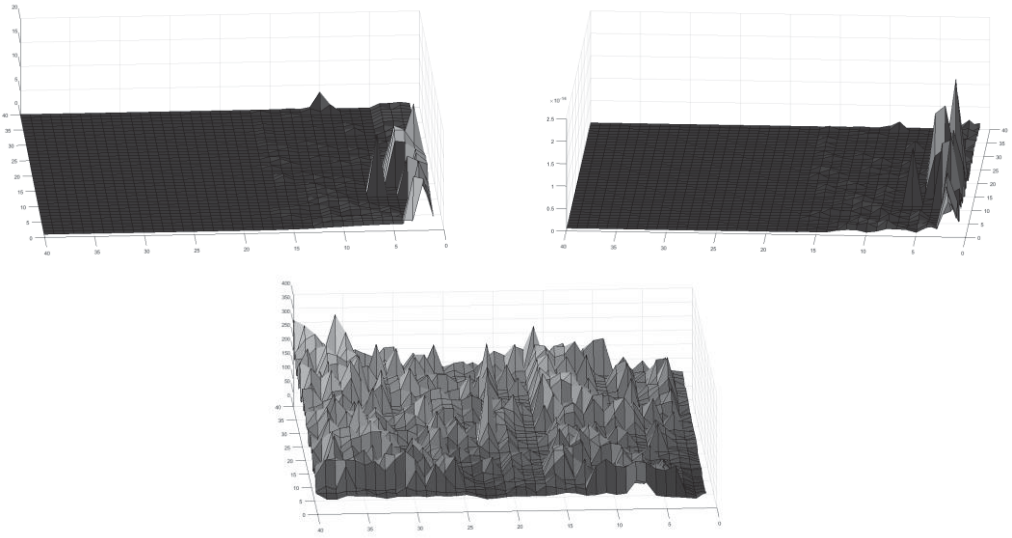

Figure 6: ARMA fit, standard deviation and run time

It can be seen that both models works well in terms of predicting the next value of the process, however the ARMA model is better. In terms of standard deviation, the ARMA model is highly superior to the NARNET model, which means that the results from the ARMA model are more reliable. In terms of run time, there is no significant difference between the two models. As a conclusion we decided to use the ARMA model, because it is more accurate and more reliable compared to the NARNET model. In terms of parameters, the 20 degree polynomials were selected, as they are accurate, reliable, and the run time is acceptable.

In the next section we analyse how the selected ARMA model behaves with different processes.

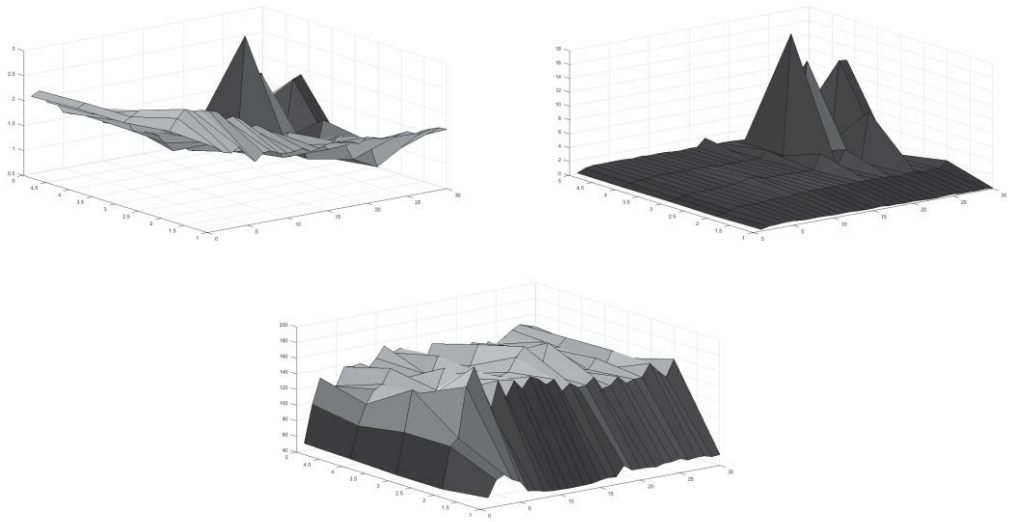

Figure 7: NARNET fit, standard deviation and run time

4.2 Analysis of the ARMA model

We evaluated our chosen Modell with 4 different tests. We tested in both normal operation and in an attack scenario the robustness of the model with different teaching and testing data length. For this test we used the Tennessee Eastman Process (Downs and Vogel, 1993).

The Tennessee Eastman Process model of Down and Vogel (Downs and Vogel, 1993) remains to be one of the most important tool for evaluating system theory concepts and validating algorithms. It is based on a real chemical process that causes the model to be a complex multicomponent system. Due to the wide usages of this process it has multiple implementations in a number of languages. For our test purposes we used a revised Matlab implementation from 2015 by Bathelt *et al* (2015).

The process consists of a reactor/separator/recycle arrangement containing two simultaneous gas-liquid exothermic reactions of the following form:

A(g) + C(g) + D(g) -> G(liq), Product 1,

A(g) + C(g) + E(g) -> H(liq), Product 2.

A sematic figure of the process looks like the following (Bathelt *et al*, 2015):

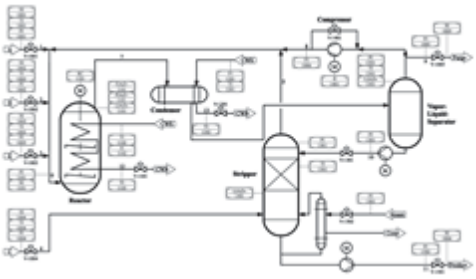

Figure 8: Tennessee Eastman process

This implementation has numerous output parameters that could have been used for model evaluation. We have chosen to use the reactor pressure parameter because of its intuitive interpretation. During multiple runs of the process pressure values were collected and ordered them into an array for the test.

The four conducted test were the following. The model was tested with different teaching intervals to measure its learning quality. After that the model was tested with different evaluation periods to test its predication quality as well.

The other two tests were to evaluate the attack detection mechanism. This time we performed the same two tests again but with known attack vectors. The results of our test are the following:

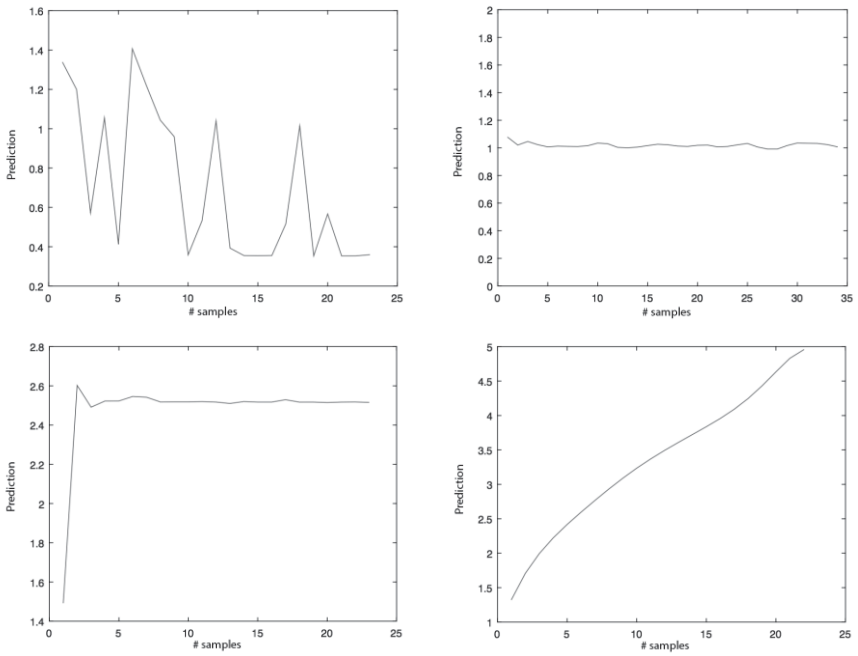

Figure 9: ARMA model analysis results

The first two figures show the results for our teaching tests. In case of a clean system, with the increase of the teaching set, the prediction gets better. It is clear that the more the model knows about the system, the better it can predict the future output. If we have at least 20 clear samples, the error of the prediction gets reasonably low.

Once we have a proper model, the increase of the prediction length makes no significant difference for the evaluation (as shown in the second figure). The ARMA model assigns the same value (indicating no discrepancy) continuously to the system.

The second set of images show the system results in case of an attack. With an increase of the teaching set the prediction results gets better. A high prediction error means that the predicted and measured values differ in a significant way. The test shows that if the clean set is long enough (at least 3-4 samples), we can detect the attack successfully.

On the last figure, our final test shows that an increase of the prediction length increases confidence of an attack prediction. This behaviour is in correspondence with our exceptions.

5. Summary

In this paper, it is shown how an intrusion detection system, which can detect attacks against industrial control systems using programmable logic controllers, was built. This solution is superior to previous approaches in terms of accuracy and usability.

According to our results, our system can detect different intrusions by identifying anomalies in the state transitions of the physical process. This approach can be used in any critical CPS, but it can be especially valuable in legacy systems, where additional security can be achieved without any modification of the system.

References

Axelsson, Stefan. Intrusion detection systems: A survey and taxonomy. Vol. 99. Chalmers University of Technology, Goteborg, Sweden: Technical report, 2000.

Bathelt, Andreas, N. Lawrence Ricker, and Mohieddine Jelali. "Revision of the Tennessee Eastman Process Model." IFAC-PapersOnLine 48.8 (2015): 309-314.

Debar, Hervé, Marc Dacier, and Andreas Wespi. "Towards a taxonomy of intrusion-detection systems." *Computer Networks* 31.8 (1999): 805-822.

J. J. Downs and E. F. Vogel. A plant-wide industrial process control problem. Computers & Chemical Engineering, 17(3):245–255, 1993.

Garitano, Iñaki, Roberto Uribeetxeberria, and Urko Zurutuza. "A review of SCADA anomaly detection systems." *Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011.* Springer Berlin Heidelberg, 2011.

Hadžiosmanović, Dina, et al. "Through the eye of the PLC: semantic security monitoring for industrial processes." *Proceedings of the 30th Annual Computer Security Applications Conference*. ACM, 2014.

Kiss, Istvan, et al. "Data clustering-based anomaly detection in industrial control systems*." Intelligent Computer Communication and Processing (ICCP), 2014 IEEE International Conference on.* IEEE, 2014.

Nardella, Davide "Snap7 package" http://snap7.sourceforge.net, 2015

Valli, Craig. "Snort IDS for SCADA Networks." (2009).

Verba, Jared, and Michael Milvich. "Idaho national laboratory supervisory control and data acquisition intrusion detection system (SCADA IDS*)." Technologies for Homeland Security, 2008 IEEE Conference on. IEEE*, 2008.

Yang, Dayu, Alexander Usynin, and J. Wesley Hines. "Anomaly-based intrusion detection for SCADA systems." *5th intl. topical meeting on nuclear plant instrumentation, control and human machine interface technologies (npic&hmit 05).* 2006.

Zhu, Bonnie, and Shankar Sastry. "SCADA-specific intrusion detection/prevention systems: a survey and taxonomy." *Proc. of the 1st Workshop on Secure Control Systems (SCS)*. 2010.