An HMM-Based Anomaly Detection Approach for SCADA Systems

Kyriakos Stefanidis[1] and Artemios G. Voyiatzis[2](✉)

[1] Industrial Systems Institute/RC ‘Athena’, Patras, Greece
stefanidis@isi.gr
[2] SBA Research, Vienna, Austria
avoyiatzis@sba-research.org

Abstract. We describe the architecture of an anomaly detection system based on the Hidden Markov Model (HMM) for intrusion detection in Industrial Control Systems (ICS) and especially in SCADA systems interconnected using TCP/IP. The proposed system exploits the unique characteristics of ICS networks and protocols to efficiently detect multiple attack vectors. We evaluate the proposed system in terms of detection accuracy using as reference datasets made available by other researchers. These datasets refer to real industrial networks and contain a variety of identified attack vectors. We benchmark our findings against a large set of machine learning algorithms and demonstrate that our proposal exhibits superior performance characteristics.

1 Introduction

The continuous interconnection of Industrial Control Systems (ICS) to public and corporate networks exposes them to the common Information Technology (IT) vulnerabilities and attacks. The security mechanisms that are traditionally used in the ICS environment cover the basic needs for authentication, authorization and (sometimes) communication confidentiality. However, they leave the Operations Technology (OT) networks open to more elaborate IT-based network attacks.

The rise of security incidents involving malicious network activity in critical infrastructures drives the need for intrusion detection technologies and mechanisms to be adapted for the OT environment. Several methodologies have been proposed recently that attempt to solve the problem of designing an efficient Network Intrusion Detection System (NIDS) specifically for the integrated IT–OT environment [21].

There are many approaches proposed in the literature, especially using anomaly detection techniques. These approaches use a combination of the known machine learning algorithms in order to determine the normal behavior of the network and detect any abnormal network traffic. The unique characteristics of network traffic in OT environments, including stable connectivity, periodicity in

© IFIP International Federation for Information Processing 2016
Published by Springer International Publishing Switzerland 2016. All Rights Reserved
S. Foresti and J. Lopez (Eds.): WISTP 2016, LNCS 9895, pp. 85–99, 2016.
DOI: 10.1007/978-3-319-45931-8_6

traffic patterns, use of standard application level protocols, are discussed in [5]. These characteristics render network-based anomaly detection a useful approach.

The main issue discussed in the literature for NIDS is the need to correlate a high enough amount of traffic (i.e., network packets) so as to decide on the abnormality of the sampled traffic. To support real-time detection of abnormal traffic, it is interesting to explore the efficiency of an approach that relies only on individual packets. Towards this direction, the use of Hidden Markov Model (HMM) approaches is limited within the literature. This despite that HMM is among the most promising approaches for this problem [19].

In this paper, we design and evaluate an HMM-based Network Intrusion Detection System (NIDS) for the OT environment. The system exploits the unique traffic characteristics of an ICS, and especially of SCADA systems communicating over TCP/IP technologies. Our aim is achieve a high-rate of detection of a wide range of attack vectors. The evaluation of the system is based on freely-available datasets that represent the operation of real industrial networks and have already been used to test several machine learning algorithms. This allows a fair comparison of our HMM-based approach with other machine learning approaches that are proposed in the literature.

The rest of the paper is organized in four sections. Section 2 provides a survey of approaches for anomaly detection in the ICS environment. Section 3 describes the proposed system architecture. Section 4 describes the experimental setup and the datasets used for evaluating the accuracy of the system. Section 5 presents and discusses our findings. Finally, Sect. 6 provides our conclusions and the future directions of work.

2 Literature Review

Many researchers tackled the problem of efficient anomaly detection in ICS environments using a wide range of machine learning techniques and targeting different application scenarios. A recent comprehensive account of such IDS, including a taxonomy of attacks is available in [21]. We revisit some representative works in the following paragraphs.

Ali *et al.* focused on the advanced metering infrastructure (AMI) [2]. The authors used a fourth-order Markov chain to model the AMI behavior via the event logs of the network. They proposed a configuration randomization mechanism in order to make the system more robust against evasion attempts.

Caselli *et al.* dealt with semantic attacks that involve sequences of permitted events and how they elude a normal NIDS [7]. The authors used discrete-time Markov chains to describe the normal operations and calculated the weighted distance between the states for the detection mechanism.

Erez *et al.* considered the Modbus/TCP protocol and focused only on the values of the control registers [9]. They classified the control registers in three classes and constructed three different behavior models, one for each class. Yoon *et al.* also considered the Modbus/TCP protocol [20]. They used Dynamic Bayesian Networks and Probabilistic Suffix Trees for traffic modeling. They also incorporates a mechanism that checks whether a detected anomaly is based on missing messages instead of an attack.

Ntalampiras *et al.* used an HMM to model the relationship between data streams from two network nodes [14]. They used a combination of emulated network components and simulated physical devices for the experimental framework. They considered only two types of attacks, namely denial of service and replay attacks, for the evaluation of the system's efficiency.

Marti *et al.* combined a segmentation algorithm with a one-class support vector machine (SVM) [12]. They constructed an anomaly detection system specifically for petroleum industry applications. Schuster *et al.* also evaluated the efficiency on one class SVMs [17].

Almalawi *et al.* explored a two-step methodology based on unsupervised learning [3]. They performed an automatic classification of normal and abnormal operation states as a first step. Then, they automatically extracted proximity detection rules for those states. The detection step is based on the kNN algorithm, raising the computational complexity to impractical levels.

Hadziosmanovic *et al.* followed the path of integrating the operational parameters of the network and its devices [10]. They extracted these parameters and used auto-regression and control limits in order to map the changes in the time domain. An alert was raised if the data did not fit the model or were outside the control limits.

Raciti *et al.* tackled the problem of designing a real-time detection system that can be implemented insider a smart metering system [16]. They used a basic clustering algorithm for this and they were more concerned on the deployment issues of such a system on a massive scale.

3 System Architecture

The proposed system is an anomaly detection system. Its underlying detection algorithm is based on Hidden Markov Model (HMM). The merits of an HMM-based approach for anomaly detection in ICS environments are discussed in [19]. This approach assumes that a model emits sequences of symbols. This model is a Markov process and its states are hidden from the observer. The model has a probability of transitioning from one state to another one and each state has a probability of emitting a symbol. An HMM is characterized by the following parameters:

- N: The number of states.
- M: The number of symbols (vocabulary).
- A: The probability distribution between stages ($N \times N$).
- B: The probability distribution to emit a symbol ($N \times M$).
- p: The probability vector on the starting state.

The overall system architecture is depicted in Fig. 1.

The system operates in two distinct phases: the *training phase* and the *detection (evaluation) phase*. In the training phase, the parameters (A, B, p) of the

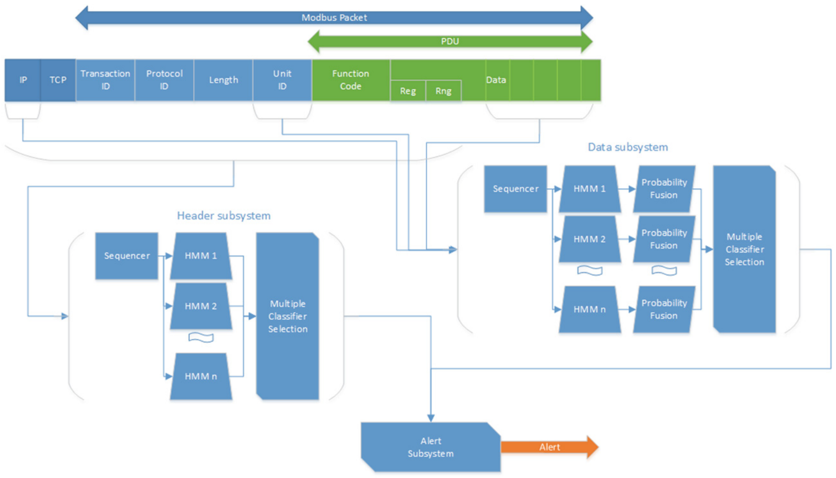

Fig. 1. System architecture

model are estimated, using a training set of sequences. The Baum-Welch algorithm is used in order to maximize the probability that the model emits the training sequences. In the detection phase, the Forward-Backward algorithm is used to calculate the probability that a received (captured) sequence is emitted by the model.

The core design problem is the selection of the sequences that are used to construct the model(s). We opted for two distinct models that are implemented in two subsystems, namely *header* and *data*. The subsystems collect different segments from each traversing network packet. The byte sequences of the segments are the sequences used at each subsystem's HMM. The exact choice of segments depends on the network and application protocols used in the SCADA system.

In general, an NIDS that targets all major ICS protocols must provide a configuration for each protocol. In the following, we will discuss the subsystems' design for the case of a Modbus/TCP protocol.

3.1 The Modbus Protocol

Modbus is the *de facto* standard for connecting industrial electronic devices in ICS. Modbus is a simple and robust protocol, with two roles (master and slave) and stateless communication of request/response frame pairs. Modbus frames can be carried over serial links or TCP/IP. The Modbus slave device is modelled as a set of four memories, namely *coils*, *discrete inputs*, *holding registers*, and *input registers*. The control loops and the reporting can be modeled as a series of *reads* and *writes* of these memories.

The format of a Modbus frame when transmitted over TCP/IP is depicted in Fig. 2. The Modbus/TCP Application Data Unit (ADU) consists of a 7-byte

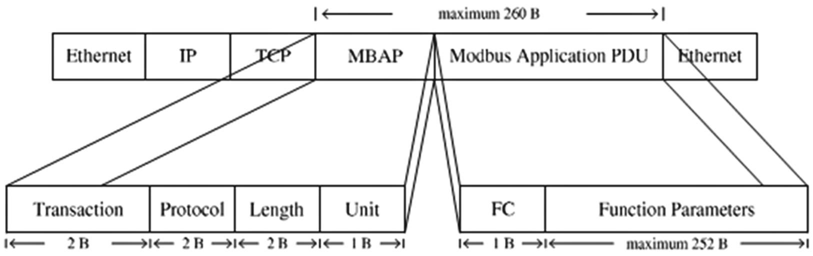

Fig. 2. Modbus frame encapsulation in TCP segments

header (Modbus Application Header, MBAP) and a Modbus Protocol Data Unit (PDU) of up to 253 bytes.

The MBAP consists of a 2-byte transaction identifier; a 2-byte protocol identifier (set to `0x0000` for Modbus); a 2-byte length field, indicating the number of the following bytes; and a 1-byte unit identifier (set to `0xFF`, equivalent to the slave address in the serial version of Modbus).

The Modbus PDU carries requests that are defined in *function codes* (FC), ranging from 0 to 127, while negative responses range from 128 to 255. Each function code can be followed by *function parameters*, passing request parameters to the slave. For example, a *read coil* function code is followed by the address (number) of the specific coil to be read. Modbus is a simple protocol and its implementation should be straightforward and bug-free; however this is not the case [18].

3.2 Header Subsystem

The first subsystem is the “Header subsystem”. This subsystem is responsible for detecting anomalies on the network traffic from attacks that manipulate the header information. Example attacks of this type include denial of service, fake devices, sending data to third parties, and Modbus response or command injection.

The header subsystem sequences the Modbus Function Code, the Modbus/TCP header, and selected parts of the TCP and IP headers. One sequence is extracted from each captured packet. The output of the subsystem is one probability.

3.3 Data Subsystem

The second subsystem is the “Data subsystem”. This subsystem outputs n distinct probabilities for each packet. The mean value of those is the final output of the subsystem for this packet. This subsystem is responsible for detecting attacks that manipulate the data that are produced by all the devices participating in the network. Such attacks are usually malicious device tampering or could be side-effects of a network attack.

The data subsystem sequences the Modbus/TCP data value(s). Each sequence is the concatenation of the IP address, the Modbus device identifier, and one data value. The sequencer extracts from each packet n sequences, where n is the number of data values in the network packet. In the case of Modbus/TCP, this is equal to the number of memory addresses requested.

3.4 Detection Process

Each subsystem contains multiple HMMs and each sequence is routed through all the contained models. The use of multiple classifiers can help reach better accuracy, as shown in [15]. The resulting probabilities from each classifier are fused together, as described in [4]. The final outcome is an overall probability for each packet. We note that all the HMMs are trained with the same training data but with different (random) starting parameters.

The last part of the detection process is the classification. Each subsystem has a certain threshold under which the packet is classified as being part of an attack. The threshold is necessary because the result of an HMM is an unweighted probability (i.e., a "log-likelihood"), which can only be evaluated in conjunction with the log-likelihoods that the HMM outputs for normal traffic.

Finally, if one of the subsystems issues an alert, the system routes this alert to the SCADA or another part of the IDS, depending on the system deployment.

4 Datasets and Experimental Setup

In order to evaluate the efficiency of the proposed system, we need a set of data from either a simulated or a real environment. This dataset should have a mix of normal traffic and attack traffic from various attacks. We opted for freely-available datasets that have already been used for the evaluation of existing systems, in order to be able to compare our approach with known solutions and algorithms.

4.1 Description of Datasets

The datasets that are originally chosen are the three datasets of [13]. These datasets contain measurements from a laboratory-scale gas pipeline, a laboratory-scale water tower, and a laboratory-scale electric transmission system. All three datasets contain pre-processed network transaction data but the lower communication layers (Ethernet, IP, and TCP) are stripped. The range on the number of entries on each dataset varies from 100,000 to 5,000,000.

Due to the vast amount of data in the given datasets, the time needed for each experiment grew prohibitively long. We resorted on using the limited versions (offered from the same source) that contain 10 % random samples from the original datasets. The effectiveness of various tested machine learning algorithms remains mostly unchanged, regardless on the version of datasets that is used for the experiment [11].

The value of these datasets for the evaluation of our system lies on the fact that these datasets contain real measurements and both the network protocol headers and the payload of each entry (measurement) are available. Furthermore, these datasets include both normal traffic and a variety of simulated attacks. Each entry is categorized in one of seven (7) attack vectors (including the normal traffic) as follows:

0. Normal (Normal)
1. Naïve Malicious Response Injection (NMRI)
2. Complex Malicious Response Injection (CMRI)
3. Malicious State Command Injection (MSCI)
4. Malicious Parameter Command Injection (MPCI)
5. Malicious Function Code Injection (MFCI)
6. Denial of Service (DoS)
7. Reconnaissance (Recon)

The attack vectors contain both header and payload manipulation patterns. We consider them sufficient for the purpose of the evaluation of our system.

One last limitation is the absence of TCP/IP data on each packet. Since our system combines the payload data, the IP address and the unit (device) identifier (`id`), we had to simplify the data sequencing subsystem and assume that the pairs (`unit id`, `measurement`) define each a distinct sensor measurement.

4.2 Training Dataset

For the construction of the training dataset, we used a subset of the entries that are classified as "normal". We excluded the entries that contain apparent erroneous measurements, in order to keep the HMM training process as accurate as possible. The erroneous measurements are probably the result of a sensor malfunction or network error. Also, it is commonly-acceptable that the training process is not performed in real time but rather off-line and based on pre-processed data. Hence, the training dataset should not contain apparently faulty packets. We also performed quantization (rounding) of the measurements in order to further enhance the accuracy of the training process for the data subsystem. The exact rounding of each value is dependent on the range of the readings for each device.

We used only a subset of the normal traffic as a training dataset, in order to measure the efficiency of our system in classifying normal data as such and evaluate the false positive probability. This traffic was not used afterwards during the detection phase.

4.3 Experimental Setup

For the experiment implementation, we used the machine learning framework Accord [1]. Accord provides a well-tested and documented library for constructing various types of HMMs, training them using the Baum-Welch learning algorithm and evaluating the log-likelihood that they generate a given sequence, using the Forward-Backward algorithm. Since the datasets were already available in text format, there was no need for actual packet capture and manipulation for the purposes of this experiment.

We note that a complete implementation of our system requires the use of an established NIDS (e.g., snort! [8]) and the implementation of the subsystems as a preprocessor that runs before the normal detection engine but once the packet is decoded.

Our initial experiments showed that the water and the gas datasets produce similar results. Therefore, we will present in the next Section only the results from the water pipeline. On the other hand, the electric data lacks information about the header values on each entry and gives only the measurements from the 132 sensors. Since our system relies on both the header and the payload data, we opted not to use this dataset in the evaluation.

5 Results and Discussion

In this section, we will present the experimental results and provide some insights regarding the use of HMMs in detecting network attacks in ICS environments. We first showcase the operation of the two subsystems and then present the overall evaluation of the system in terms of detection efficiency. Then, we present a comparison of the results against already established algorithms that have been tried on the same dataset.

5.1 Operation Scenarios

Figure 3 depicts a time window during which an NMRI attack takes place. One of the outcomes of this attack is that the sensor measurements are skewed to the extent that they cross the boundaries of a normal operation. In the first part of the Figure, the sensor measurements as taken from the dataset are shown; the effect of the attack is clearly identifiable between the samples 36 and 67. In the second part of the Figure, the estimations by the data subsystem are depicted as a series of likelihoods for each entry (the exponent of the log-likelihood that the subsystem outputs). The likelihood becomes zero during the attack. This piece of information can be used as an alert signal from the data subsystem, as it correctly identifies one of the outcomes of the attack.

It might be the case that the effect of this NMRI-type attack, i.e., the skewing of the measurements, is an unintended consequence of the way the dataset was compiled in first place. We note that (i) only few of the attacks in the dataset have any effect on the actual sensor measurements and (ii) we evaluate the effectiveness of *both* the subsystems in our system. Thus, we include this data skewness as an attack indicator for the purpose of the experiments. Such manipulations of the readings are common in attack scenarios where an outsider tampers the information in-transit rather than directly in the sending system.

Figure 4 depicts a time window during which a Recon attack takes place. While this attack is active, various header fields deviate from their "normal" values, as the attacker tries to derive useful information. The first part of the Figure

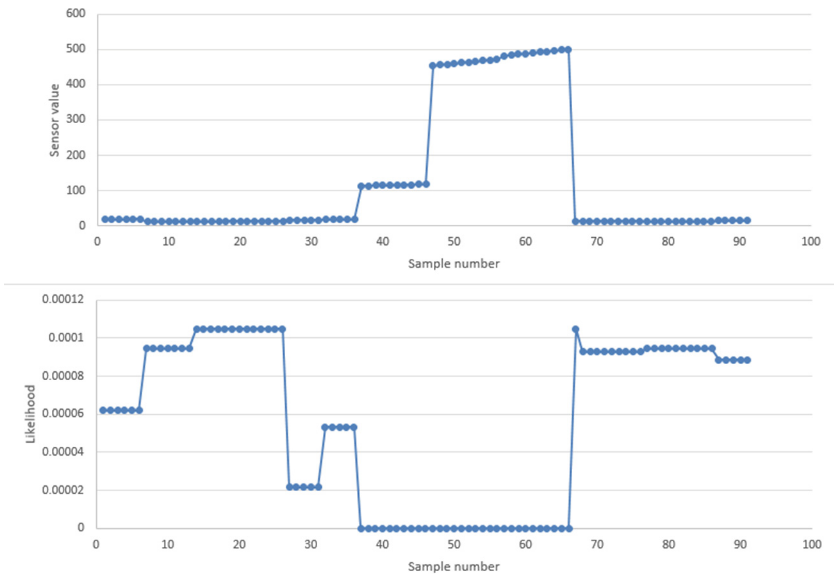

Fig. 3. Attack and detection under an NMRI attack

depicts five different header fields that should either stay constant or change in a predictable way (e.g., `command_address` and `response_address` should change together). During the attack, they get abnormal values. In the second part of the Figure, the estimations by the header subsystem are provided. During attack (samples 11–16; 18; 20; and 34–48), the likelihood drops to zero in almost all cases. Again, we use this piece of information as an alert signal from the header subsystem.

We note that for the case of the header values, the likelihoods during normal operation are more or less stable, while for the case of the data values (e.g., those shown in Fig. 3), the likelihoods exhibit a wider distribution. However, in both cases the likelihood drops to almost zero during the attacks in both cases. Still, there are scenarios where the likelihood does fall a few orders of magnitude compared to normal traffic but does not reach zero. These facts affect the selection of a threshold value for denoting an abnormal behavior. The threshold for the data subsystem needs to be set after the training procedure and upon careful examination of the system's output (likelihoods) during normal operation.

5.2 Results

Classification Efficiency. The efficiency of the system in correctly classifying normal and attack traffic for the whole dataset is summarized in Table 1. Line one and two depict the actual number of entries for each attack vector (0–7) and the number of entries that are correctly classified. Line three depicts the classification efficiency as a percentage for each attack vector. The "All" column contains the

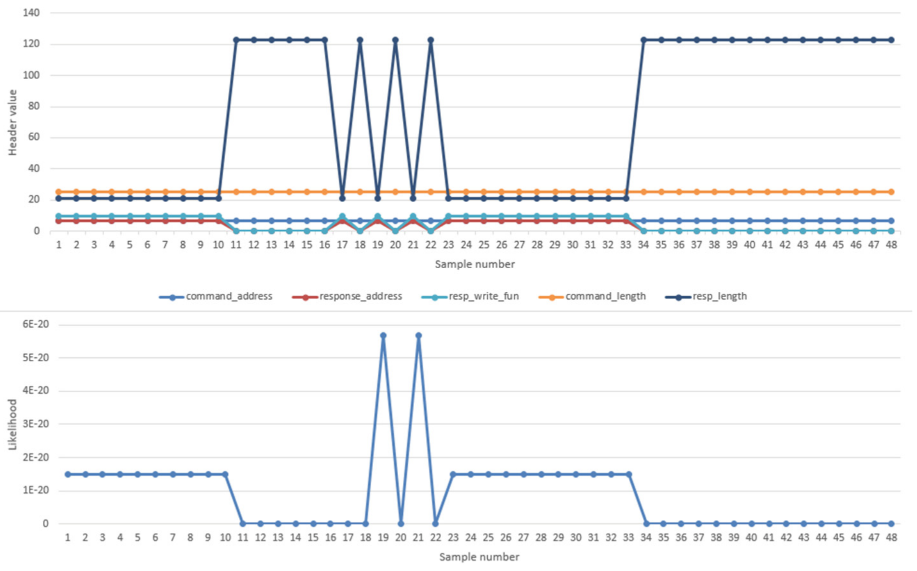

Fig. 4. The attack and detection graph under a Reconnaissance attack

overall classification efficiency for all the entries (normal and attacks), while the "Attacks" column contains the classification only for the attack entries.

The classification efficiency is more than 95 % for most of the attacks and the normal traffic. The classification does not succeed in identifying the CMRI and MSCI attacks. Upon further inspection, it appeared that this can be attributed to some flaws in the training dataset that allowed the HMMs to be trained to recognize such attacks as normal traffic. However, these flaws were identified by the header subsystem; the data subsystem could not identify them as the attack forced data values that are predictable but yet within the "normal" boundaries for the specific sensor they came from. Further details as well as the ramifications of this behavior are discussed in the next section.

There are only a few false positives, i.e., normal traffic classified as attack traffic, less than 1 %. The false positives consist almost entirely of erroneous sensor readings or network packets. This is an expected outcome, given the preprocessing of the training dataset. It can be considered as a trade-off between raising the probability of correctly identifying actual attacks and raising the probability of generating false positives.

Comparisons. The dataset we used for the evaluation of our system has been used extensively as a reference dataset for evaluating a wide range of machine learning algorithms [11]. The applicability of these algorithms for anomaly detection in the ICS environment is studied in [6]. We compare the efficiency of our approach with the published study in Table 2. While not all the algorithms are

Table 1. Detection accuracy

Classification	Normal	NMRI	CMRI	MSCI	MPCI	MFCI	DoS	Recon	All	Attacks
Actual	19,503	1,198	1,457	209	410	155	135	4,132	27,199	7,696
Detected	19,424	1,159	0	0	389	155	134	4,132	25,393	5,969
Percentage (%)	99.6	96.7	0.00	0.00	94.9	100	99.3	100	93.4	77.6

directly comparable with our approach, the efficiency of the latter is directly comparable to most of them.

In the case of a DoS attack, 16 out of the 34 algorithms have less than 90 % accuracy, while our approach reaches 99 %. During Recon, MPCI, and MFCI attacks, almost all algorithms produce nearly perfect results. The case of CMRI and MSCI is an area of future improvement for our system, while they can be handled by most the evaluated algorithms. In contrast, the NMRI attack is a hard case for 14 algorithms, while it can be easily detected by our approach. All the algorithms and our approach as well exhibit a very low number of false positives in all cases.

It is clear that our approach gives comparable results to the most effective machine learning algorithms while using only the information on each individual sample without the need of correlation of multiple samples of the same traffic flow. It is also clear that it is unable to detect all the attack vectors within the dataset. A discussion on why those attack vectors eluded detection and how this can be remedied is found in the next section.

5.3 Discussion

The approach we propose exploits the unique characteristics of an ICS environment and is able to identify most of the attacks presented in the reference datasets. The detection rate is comparable to already-tested machine learning algorithms and through careful adjustment of the training data and the HMM parameters, this rate can be further enhanced. As an example, the detection rate for the NMRI attack has doubled by eliminating easily identifiable erroneous data from the training set. Also, system design parameters, such as the number of parallel HMMs and the number of hidden states can affect the efficiency of the overall system.

We highlight that our proposed approach relies only on per-packet information and needs not to correlate multiple packets on the same network stream. This makes the system suitable for real-time applications, where correlation of multiple packets can adversely effect the speed of detection. It also allows it to be deployed in data storage constrained environments due to non existent storage requirements. Considering also the case of low rate (or stealthy) attacks, the system is able to detect such an attack without the need of a large number of packets that are part of the attack.

On the topic of the training data, we see that the proposed system exhibits increased sensitivity to errors or malicious activities during the preparation of

Table 2. Comparisons with known machine learning algorithms (as reported in [11])

Algorithm	Normal (0)	NMRI (1)	CMRI (2)	MSCI (3)	MPCI (4)	MFCI (5)	DOS (6)	Recon (7)
Proposed and Evaluated system	100 %	97 %	0 %	0 %	95 %	100 %	99 %	100 %
Best First Decision Tree (BFTree)	100 %	97 %	99 %	87 %	99 %	95 %	95 %	100 %
Decision Stump	99 %	0 %	0 %	0 %	0 %	0 %	0 %	100 %
FaultTree (FT)	100 %	94 %	100 %	93 %	99 %	95 %	96 %	100 %
J48 Decision Tree	100 %	95 %	100 %	90 %	99 %	73 %	91 %	100 %
J48Graft Decision Tree	100 %	95 %	100 %	89 %	99 %	68 %	88 %	100 %
Logiboost Alternating Decision Tree (LADTree)	100 %	94 %	99 %	93 %	99 %	0 %	73 %	100 %
Logistic Model Tree (LMT)	100 %	86 %	100 %	93 %	99 %	95 %	93 %	100 %
Logistic Regression	100 %	4 %	99 %	93 %	99 %	95 %	67 %	100 %
Multilayer Perceptron	98 %	2 %	99 %	93 %	99 %	95 %	68 %	100 %
Nave Bayes Tree (NBTree)	100 %	96 %	99 %	93 %	98 %	95 %	95 %	100 %
Radial Basis Function Network (RBFNetwork)	98 %	1 %	99 %	93 %	99 %	95 %	88 %	100 %
RandomErrorPruning Tree (REPTree)	100 %	95 %	100 %	90 %	99 %	95 %	95 %	100 %
RandomForrest	100 %	96 %	100 %	90 %	99 %	93 %	93 %	100 %
RandomTree	99 %	96 %	100 %	90 %	98 %	81 %	91 %	100 %
SimpleCart	100 %	96 %	100 %	88 %	99 %	95 %	94 %	100 %
SimpleLogistic	98 %	36 %	99 %	93 %	99 %	95 %	68 %	100 %
Sequential Minimal Optimization (SMO)	98 %	1 %	99 %	93 %	99 %	73 %	44 %	100 %
BayesNet	98 %	98 %	95 %	97 %	100 %	100 %	99 %	100 %
ComplementNaiveBayes	100 %	0 %	0 %	0 %	29 %	100 %	0 %	100 %
DMNBtext	100 %	0 %	0 %	0 %	44 %	100 %	0 %	100 %
NaiveBayes	43 %	0 %	99 %	97 %	99 %	100 %	96 %	100 %
NavieBayesMultinomial	100 %	0 %	0 %	97 %	81 %	100 %	39 %	100 %
NaiveBayesMultinomial Updateable	100 %	0 %	0 %	97 %	81 %	100 %	39 %	100 %
NaiveBayesSimple	0 %	95 %	98 %	56 %	62 %	0 %	0 %	2 %
NaiveBayesUpdateable	43 %	0 %	99 %	97 %	99 %	100 %	96 %	100 %
ConjunctiveRule	100 %	0 %	0 %	0 %	0 %	0 %	0 %	100 %
DecisionTable	98 %	98 %	95 %	94 %	98 %	95 %	91 %	100 %
DTNB	98 %	98 %	95 %	97 %	99 %	100 %	100 %	100 %
Jrip	99 %	98 %	94 %	97 %	99 %	100 %	100 %	100 %
Nnge	97 %	97 %	75 %	97 %	99 %	100 %	97 %	100 %
OneR	97 %	98 %	95 %	0 %	0 %	0 %	0 %	100 %
PART	99 %	98 %	95 %	97 %	99 %	100 %	100 %	100 %
Ridor	99 %	98 %	94 %	97 %	99 %	100 %	100 %	100 %
ZeroR	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %

the training data. As an example, our system was unable to detect the CMRI attack; this can be mostly credited to the fact that a limited amount of normal data exhibited the same pattern, probably due to some network error. This fact remained unnoticed during the preparation phase of the training set, which in turn resulted in the less-optimal training of the system's parameters. Hence, we highlight the importance of having access to carefully-inspected and validated reference datasets.

The reference datasets we used for our evaluation exhibit some limitations that might affect the applicability of our approach in real-world scenarios. Such limitations include the lack of information for the lower-than-IP network layers and that the measurements come from only one sensor node. We hope that more detailed datasets become publicly available for the benefit of the research community and the industry practice alike. We consider as an interesting enhancement for the detection itself to incorporate information regarding the actions performed by the SCADA operators themselves; such information can be utilized to detect suspicious traffic, which is a result of a malicious operators' actions.

As it has been raised many time in the research literature already, an anomaly-based NIDS *cannot* effectively capture the whole picture of the complex ICS system under attack. It is better to integrate it in a defense-in-depth strategy, combined with a signature-based IDS for known attacks; this would speed up more the detection capabilities and provide more fine-grained information under specific attacks. Furthermore, this integration could significantly reduce the number of fake alerts (false positives) for the SCADA operator and contribute towards a more comprehensive reporting system that can present the alerts in a meaningful for the SCADA operator context.

6 Conclusions and Future Work

In this paper, we presented a novel approach for an HMM-based NIDS. Our approach exploits the unique characteristics of the ICS environment. We evaluate the efficiency of our approach using reference datasets taken from a real-world industrial system. These datasets contain a diverse set of attack vectors allowing for the evaluation of our approach under realistic operation scenarios.

The proposed system succeeds in detecting most of the attack vectors, both targeting application logic via payload information (i.e., the sensor measurements) and network logic via header information. We evaluated the detection rate and compared the efficiency of our approach with a large set of algorithms available in the literature. The detection rate is directly comparable with the latter and can further improved by tuning the system parameters and the quality of the training set.

More importantly, the proposed system, in contrast to other approaches, can produce the results on a per-packet basis, without needing to correlate samples with historical traffic information. This makes it more suitable for real-time applications and high-speed or large-scale environments (high traffic volumes). Also, for detecting low-rate, stealthy attacks, that are often used as part of advanced persistent threats (APTs) against critical systems and infrastructures.

An interesting direction for future work includes the integration, if possible, of information related to actions performed by SCADA operators that can result unexpected (anomalous) traffic. Also, the integration of our approach with a signature-based IDS. Also, a study on the impact of the basic design parameters (number of parallel HMMs, number of hidden states, etc.) on the overall performance of the system is a mandatory step towards its adoption and deployment in real-time critical ICS. Last but not least, we consider necessary to extend the scope of the experiments using more diverse datasets, if they become publicly-available, which should incorporate a larger ICS environment, including more network segments, ICS devices, and sensors.

Acknowledgements. This work was partially supported by the GSRT Action "KRIPIS" of Greece with national and EU funds in the context of the research project "ISRTDI" and the COMET K1-Centres programme line of the Austrian Research Promotion Agency (FFG).

References

1. Accord.NET: Accord.NET Machine Learning Framework (2016). http://accord-framework.net/
2. Ali, M.Q., Al-Shaer, E.: Randomization-based intrusion detection system for advanced metering infrastructure. ACM Trans. Inf. Syst. Secur. **18**(2), 7:1–7:30 (2015). http://doi.acm.org/10.1145/2814936
3. Almalawi, A., Yu, X., Tari, Z., Fahad, A., Khalil, I.: An unsupervised anomaly-based detection approach for integrity attacks on SCADA systems. Comput. Secur. **46**, 94–110 (2014). http://dx.doi.org/10.1016/j.cose.2014.07.005
4. Ariu, D., Tronci, R., Giacinto, G.: HMMPayl: An intrusion detection system based on Hidden Markov Models. Comput. Secur. **30**(4), 221–241 (2011). http://dx.doi.org/10.1016/j.cose.2010.12.004
5. Barbosa, R.R.R.: Anomaly detection in SCADA systems: a network based approach. Ph.D. thesis, University of Twente, Enschede, April 2014. http://doc.utwente.nl/90271/
6. Beaver, J.M., Borges-Hink, R.C., Buckner, M.A.: An evaluation of machine learning methods to detect malicious SCADA communications. In: 2013 12th International Conference on Machine Learning and Applications, vol. 2, pp. 54–59 (2013). http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6786081
7. Caselli, M., Kargl, F.: Sequence-aware intrusion detection in industrial control systems. In: Proceedings of the 1st ACM Workshop on Cyber-Physical System Security, CPSS 2015, pp. 13–24 (2015)
8. Cisco: Snort (2015). https://www.snort.org/
9. Erez, N., Wool, A.: Control variable classification, modeling and anomaly detection in Modbus/TCP SCADA systems. Int. J. Crit. Infrastruct. Prot. **10**, 59–70 (2015). http://linkinghub.elsevier.com/retrieve/pii/S1874548215000396
10. Hadžiosmanović, D., Sommer, R., Zambon, E., Hartel, P.H.: Through the eye of the PLC: semantic security monitoring for industrial processes. In: Proceedings of the 30th Annual Computer Security Applications Conference, ACSAC 2014, pp. 126–135 (2014). http://dl.acm.org/citation.cfm?id=2664243.2664277

11. Hsu, J., Mudd, D., Thornton, Z.: Mississippi state university project report - SCADA anomaly detection project summary. Technical report, Mississippi State University (2014). http://www.ece.uah.edu/thm0009/icsdatasets/MSU_SCADA_Final_Report.pdf
12. Martí, L., Sanchez-Pi, N., Molina, J., Garcia, A.: Anomaly detection based on sensor data in petroleum industry applications. Sensors **15**, 2774–2797 (2015). http://www.mdpi.com/1424-8220/15/2/2774/
13. Morris, T., Srivastava, A., Reaves, B., Gao, W., Pavurapu, K., Reddi, R.: A control system testbed to validate critical infrastructure protection concepts. Int. J. Crit. Infrastruct. Prot. **4**(2), 88–103 (2011). http://www.sciencedirect.com/science/article/pii/S1874548211000266
14. Ntalampiras, S., Soupionis, Y., Giannopoulos, G.: A fault diagnosis system for interdependent critical infrastructures based on HMMs. Reliab. Eng. Syst. Saf. **138**, 73–81 (2015). http://dx.doi.org/10.1016/j.ress.2015.01.024
15. Perdisci, R., Ariu, D., Fogla, P., Giacinto, G., Lee, W.: McPAD: a multiple classifier system for accurate payload-based anomaly detection. Comput. Netw. **53**(6), 864–881 (2009). http://dx.doi.org/10.1016/j.comnet.2008.11.011
16. Raciti, M., Nadjm-Tehrani, S.: Embedded cyber-physical anomaly detection in smart meters. In: Hämmerli, B.M., Kalstad Svendsen, N., Lopez, J. (eds.) CRITIS 2012. LNCS, vol. 7722, pp. 34–45. Springer, Heidelberg (2013)
17. Schuster, F., Paul, A.: Potentials of using one-class SVM for detecting protocol-specific anomalies in industrial networks. In: 2015 IEEE Symposium Series on Computational Intelligence, pp. 83–90 (2015)
18. Voyiatzis, A., Katsigiannis, K., Koubias, S.: A Modbus/TCP fuzzer for testing internetworked industrial systems. In: 20th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2015), Luxembourg, 8–11 September 2015
19. Yasakethu, S., Jiang, J.: Intrusion detection via machine learning for SCADA system protection. In: The 1st International Symposium for ICS & SCADA Cyber Security Research, pp. 101–105 (2013)
20. Yoon, M.k., Ciocarlie, G.F.: Communication pattern monitoring: improving the utility of anomaly detection for industrial control systems. In: NDSS Workshop on Security of Emerging Networking Technologies (SENT) (2014)
21. Zhu, B.X.: Resilient control and intrusion detection for SCADA systems. Ph.D. thesis, EECS Department, University of California, Berkeley, May 2014. http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-34.html