Intrusion Detection for sequence-based attacks with reduced traffic models

Benedikt Ferling¹, Justyna Chromik², Marco Caselli³, and Anne Remke^{1,2}

¹ Westfälische Wilhelms-Universität Münster, Germany
{anne.remke}@uni-muenster.de

² University of Twente, Enschede, The Netherlands
{j.j.chromik, a.remke}@utwente.nl

³ Siemens AG, München, Germany
{marco.caselli}@siemens.com

Abstract. Securing control networks (e.g. for power and gas distribution) requires dedicated approaches. Sequence-aware intrusion detection models the network traffic under normal operation to identify malicious behavior. Unfortunately, such models are often large and difficult to handle. This paper proposes a method that generates smaller traffic models and discusses the accuracy of those reduced models in the context of a real control infrastructure employing the IEC 60870-5-104 protocol.

1 Introduction

Supervisory Control and Data Acquisition (SCADA) systems are used to monitor critical infrastructures [1], by automating communication and control of, e.g., power distribution. As shown by the US grid hack [2] and the Ukrainian grid hack [3], both causing power outages that lasted for several hours, SCADA systems are vulnerable and can cause serious damage if not properly secured.

Stuxnet [4] has demonstrated that hackers can strike critical infrastructures by directly hitting the physical process misusing internal process-knowledge. Attacks of this kind are commonly known as *semantic attacks* and are usually of higher complexity compared to standard cyber-attacks. Among all semantic attacks, *sequence attacks* can strike the infrastructure by just misplacing perfectly legal messages within a communication stream [5].

Traditional Intrusion Detection Systems (IDS) identify malicious traffic in different ways: whitelisting [6], stateful approaches [7,8], and specification-based approaches [9] exist. In contrast, we focus on the detection of so-called sequence attacks, which employ perfectly legal messages arranged in an unforeseen order and make it difficult for an IDS to detect the malicious activity.

Sequence-based intrusion detection relies on the regularity of traffic that is present in SCADA traffic and does not fit networks with large traffic variety, e.g. the Internet. However, industrial control systems show regular and consistent communication patterns [10], which can be used e.g. for anomaly detection [11]. Hence, analyzing the communication using a probabilistic state-based approach,

which keeps track of message ordering, is a solution to identify sequence attacks in control networks [12].

Sequence-based and state-based approaches close to the presented work can be found in [5] and [13], respectively. However, such analysis may be time-consuming, depending on the complexity of the communication within the network and hence, the size of the resulting models. This paper aims to reduce this complexity, while keeping the detection accuracy for most types of sequence attacks. The approach shown in [5] may result in large Discrete-time Markov Chains (DTMCs), which however contain states which represent almost identical system states. We change the generation algorithm, such that the resulting models are considerably smaller, leading to lower computation times, as well. This is done by combining states with overlapping information or by abstracting from specific packet information. We investigate the accuracy of the resulting models by performing 9 different experiments on real traffic from a Dutch gas facility which uses the protocol IEC 60870-5-104, also known as *IEC-104* [14]. IEC-104 is widely used in industrial system control, especially in the context of energy grids [15].

The protocol supports the use of *Information Objects* (IOs) to classify data. Individual packets may transfer several Information Objects, addressed using *Information Object Addresses* (IOAs). This possibly results in ranges of IOAs per packet. This paper shows that combining states that represent (almost) similar packets with overlapping IOAs and abstracting from the IOAs altogether results in much smaller traffic models. Clearly, when comparing real traffic with the previously developed traffic models, smaller models will result in smaller computation times. Our case study shows that reduced models may also decrease the number of false positive alerts.

The paper is organized as follows. Section 2 explains the basics of SCADA and IEC-104. Section 3 explains the reduction methods for DTMCs and our approach is validated in Section 4. Section 5 concludes the paper.

2 SCADA systems

This section briefly introduces SCADA systems and the IEC-104 protocol.

2.1 SCADA

SCADA systems connect *field stations*, which directly control the physical process and the *control room*, that monitors the state of the system, as shown in Figure 1. The *field station* hosts sensors, actuators and Programmable Logic Controllers (PLCs). *Sensors* measure and transmit data of the monitored process over serial wires to PLCs. *Actuators* influence the systems behavior, by, e.g., opening or closing a switch connecting a power line. *PLCs* collect the field data and send it to the data acquisition server via the communication system. They also receive commands from the control network, which are then executed on

Fig. 1. SCADA in power distribution

Fig. 2. Rail switching system

the appropriate actuator. In the *control room*, data of the field network is evaluated and appropriate actions are triggered. The *data acquisition server* collects, stores and distributes process data.

2.2 IEC-104 Protocol

The IEC-104 protocol is used for communication between field stations and control room [15]. It operates on top of TCP/IP. An IEC-104 packet consists of the *application protocol control information* (APCI) and the *application service data units* (ASDU) which together are called the *application protocol data unit* (APDU). APCI determines whether the packet has U-, S- or I-format. U-format initiates and terminates sessions between two devices, while the S-format acknowledges the received data. We focus on the I-format, which contains relevant information in the ASDU fields. The most important fields for our work are:

- *Type identification* (*TypeId*) identifies a function that the device should execute, e.g., 103 $\rightarrow$ *clock synchronization command* or 102 $\rightarrow$ *read command*.
- *Number of objects* defines the number of *information objects* found in the ASDU. This can be more than one. The *TypeId* is assigned to all these information objects inside of that packet.
- *ASDU address fields* (*ASDU-Address*) contains the address which all objects of the ASDU refer to.
- *Information object address fields* (*IOA*) refers to a specific information object, such as a reading from an element (voltage, current), a state of an element (on/off), or threshold setting. A packet may have up to 127 information objects. Multiple IOAs may come within the same response, e.g., IOAs from 1-5, and IOAs from 11-13.

While each vendor has its own implementation, the general structure follows the specification in [16] and [17].

2.3 SCADA sequence attacks

This section explains the concept of sequence attacks and provides an example inspired by the large power outage in North Holland on March 27th 2015 [18]. Sequence attacks are specific to industrial control systems and potentially harm a system by sending valid messages or commands, which are misplaced or out-of-order [5]. To take control of the process, an attacker can either reprogram a PLC or directly control the process from the network, e.g., by taking control over the communication channel. When controlling the process, an attacker sends commands in an order or timing not intended by the process. This potentially harmful sequence of commands is then called sequence attack [5].

The concept of *Interlocks* describes constraints on the execution of commands. In power distribution, this applies e.g. to the order in which power switches and disconnectors are used. According to IEC 60947-3, whenever a power line has to be disconnected, first, the power switch disconnects, which turns off the current on the power line. Only then, the disconnector is used to physically isolate the power line. Otherwise, a potentially dangerous electric arc is created. In order to connect a power line, first, the disconnector has to be connected, before the switch is closed.

Figure 2 shows rails A and B, which are used interchangeably depending on the switches' configuration. The disconnector on rail A and the switch are closed, while the disconnector on rail B is open. Hence, the power line at the bottom is connected to rail A. To switch from rail A to B, first the switch opens, then the disconnector on rail A opens. Next, the disconnector on rail B closes and then the switch closes.

In case of the outage that happened in the North Holland, the disconnector did not entirely connect before the power switch was turned on. This caused a short circuit, which resulted in a power outage of several hours for more than a million households and disruptions on one of the major European airports [19]. Although the incident was caused by a technical and human error, a similar situation would arise by sending legitimate commands in the wrong order to the PLC controlling the switches and disconnector. Although most of the PLCs do check said interlock internally, some operators perform this check at the central control room. Hence, if an attacker gains control over the communication channel to the remote PLC directly, the constraint check will not be performed. Even if interlocks are properly configured, any attempt to switch in a wrong order should be reported to the operator and a sequence-based IDS would fit the purpose.

3 Message sequences

We explain the IEC-104 traffic model, sequence attacks and the performed reductions.

3.1 Representing traffic sequences as DTMCs

Following the approach presented in [5] traffic is represented as a sequence of exchanges in terms of a Discrete Time Markov Chain (DTMC). In the following

Data: Sequence of Events

Result: DTMC representing the sequence of events

```

1 for all  $e_{t_n} \in \text{sequence}$  do
2    $State_{DTMC} \leftarrow \text{extractAttributes}(e_{t_n});$ 
3   if  $State_{DTMC} \in DTMC$  then
4      $\text{update}(State_{DTMC}, DTMC);$ 
5   else
6      $\text{add}(State_{DTMC}, DTMC);$ 
7   end
8   if  $Transition_{previousState, State_{DTMC}} \in DTMC$  then
9      $\text{update}(Transition_{previousState, State_{DTMC}});$ 
10  else
11     $\text{add}(Transition_{previousState, State_{DTMC}}, DTMC);$ 
12  end
13   $previousState \leftarrow State_{DTMC}$ 
14 end

```

Algorithm 1: DTMC modeling of sequences as in [5]

we use DTMC that can be defined as a tuple $\mathcal{M} = (S, T)$, where S is a finite set of states and T is a transition relation, that assigns probabilities to states $(s_1, s_2) \in T$, i.e. if there exists a transition between the states s_1 and s_2 . Note that the sum of the outgoing transition probabilities per state always equals one.

States in the DTMC reflect the kind of communication that takes place. Hence an event in the sequence, i.e., the transmission of a packet, is associated with a state. Transitions model the choice between successor packets together with the respective probability of this event taking place. We transform the network traffic traces into time-ordered list of events. This paper only considers the communication between two devices and therefore an *event* e_{t_n} , which takes place at time point $t_n \in \mathbb{R}^+$ is defined as a triple $\langle \text{Direction}, \text{Address}, \text{Service} \rangle$, which takes values from the IEC-104 specification, as follows:

- **Direction** either takes the value ‘request’ or ‘response’,
- **Address** contains the ‘ASDU-address’ and the ‘IOAs’,
- and **Service** is the ‘Type-Id’ that identifies a function.

A *sequence* (l) sorts events according to their time of occurrence from old to new. It is defined as a time ordered list of events e_{t_n} , such that $t_n < t_{n+1}$ for $n \in \mathbb{N}$. Alg. 1 then builds a Discrete Time Markov Chain traffic model from a sequence of events, abstracting from possibly different inter-event times, in the following five steps:

- S1** loops over all events in the sequence (c.f. Alg. 1 l.1-14), processing its events.
- S2** extracts the attributes of an event and stores them in variable ‘ $State_{DTMC}$ ’ (c.f. l. 2). The state to which an event leads is defined by: Request, Response, ASDU-Address, IOAs, TypeId and the ‘control field format’.
- S3** checks whether that state is already present in the DTMC. Then the counter indicating how often that state has been visited is increased in the corresponding state. Otherwise, a new state is added to the DTMC in line 6.

Fig. 3. Scenario with single switch and disconnector

- S4** updates the transitions. If the transition to ‘ $State_{DTMC}$ ’ is part of the DTMC the transition probabilities and the transition counter are updated (line 9). If the transition is new, line 11 adds a transition from ‘previousState’ to ‘ $State_{DTMC}$ ’ to the DTMC.
- S5** updates the variable ‘previousState’ in line 13 with the value created in line 2.

3.2 Sequence attacks

This section describes sequence attacks detected by comparison with a DTMC built on benign traffic. For example, reconsider the combination of switch and disconnector (c.f. Section 2.3), as presented in Fig. 3. Legal commands for the switches are ‘open’ and ‘close’. Let the initial state of the system be an open switch and a closed disconnector.

The DTMC traffic model has four states: $S = \{s_1, s_2, s_3, s_4\}$, where

- s_1 models that a command to open the switch is sent,
- s_2 opens the disconnecter,
- s_3 commands to close the disconnecter, and
- s_4 requires to close the switch.

Those states and the corresponding transition probabilities are shown in Fig. 3. As discussed before, a command to open the disconnecter should always be preceded by the corresponding command to open the switch. Vice-versa, the disconnecter should always be closed before the switch is closed. The only bidirectional transitions between states are between open and close disconnecter and open and close switch. While it may occur that a switch is, e.g., immediately closed after being opened, this will not occur often and hence has a low transition probability. Furthermore, each state is equipped with a self-loop that happens with a relatively low probability. This corresponds to the same command being sent multiple times, which can legally happen, e.g., due to packet retransmission. Recall that IEC-104 runs on top of TCP, which can cause retransmissions due

to preliminary timeouts or the loss of acknowledgements. In the following we distinguish between three types of violations:

1. **New transition violation** stems from a valid packet that is however not expected in the sequence of commands [5]. Consider the DTMC is in state s_4 of the switch example, i.e., the last packet contained the command to close the switch. If the next command would request to open the disconnecter a *new transition violation* is encountered, as the DTMC model does not contain a transition that corresponds to this sequence of commands, namely *close switch* succeeded by *open disconnecter*. An alert would be issued to the operator in this case to warn about a potential intrusion. Fig. 4 shows the violation as a red dashed line.
2. **New state violation** occurs when an unexpected command is sent to the controller. In our example, the switch could receive the command to change some threshold value. This is a legitimate command, which however does not occur often and has not been part of the traffic used to train the DTMC. Hence, there is no state that corresponds to this command. Fig. 5 shows the violation as a red dashed line and state.
3. **Anomalous transition frequency**, a so-called timing-violation occurs when a single transition is used too often. The commands arrive in an expected order, however they occur with a probability that deviates from the transition probability more than a certain predefined threshold. To achieve this, a parallel DTMC model is trained with the current sequence and compared to the previously trained model after each event. For example if the switch is opened and closed repeatedly, the transition probability between *open switch* and *close switch* will grow to exceed the transition probability of 0.1 in the originally trained DTMC. Hence, an anomaly is observed, and an alarm is issued, which could prevent the hardware from being harmed. Fig. 6 shows the DTMC trained from the current sequence. The transition probabilities that differ from the original DTMC are indicated as red dashed lines.

3.3 DTMC reduction

The traffic models that result from applying Algorithm 1 to realistic scenarios can be very large, as shown in [17]. However, many states of the model differ for just a state parameter [5]. In the case of IEC-104 traffic we can leverage this by combining states with overlapping IOAs and by completely abstracting from the information contained in IOAs. Consider a traffic capture where the first message is a server request asking for *reading* IOs with addresses 1-10, in the second message it requests readings from addresses 6-10, in the third message it asks for addresses 1-5, and in the last message it reads from address 125. The original approach explained in [5] would create four different states for these four different read requests. Using the original implementation, we noticed that many new states appear because a request is sent to a different subset of IOAs.

Fig. 4. DTMC with a transition violation

Fig. 5. DTMC with a state violation **Fig. 6.** DTMC with a probability violation

We therefore compare DTMCs built according to Section 3.1 without reduction with two types of reduced DTMCs, namely, (i) where states with *overlapping* IOAs are merged and (ii) where information contained in IOAs is not taken into account for differentiating states. While the reference case corresponds to the approach in [5], the first reduction case relies on the observation that the SCADA server asks for various IOAs, although the function (*TypeId*, c.f. Section 2.2) remains the same. In the example above, the first reduction approach corresponds to creating two states instead of four: one for IOAs 1-10, and a second for IOA 125. The second reduction case presents a more radical reduction approach, which completely abstracts from the IOAs and only takes the information ‘Direction’, ‘Service’ and ‘ASDU address’ into account.

For some functions like the *read command*, the process does not suffer if the reading is performed at a different place in the sequence, and merging all IOAs would greatly reduce the sizes of the DTMCs. In the mentioned example, this would mean that all reading commands refer to a single state.

We have generated traffic models from a SCADA trace obtained at a Dutch gas facility, and in the following we show the resulting DTMCs for the two private IP addresses 172.31.1.100 and 172.31.8.170. The trace consists of 10 days of traffic captured in 2011. All traffic models presented in this section have been generated using the entire available traffic capture using Algorithm 1, which

Fig. 7. Original DTMC

has been changed slightly to implement also the reductions *overlapping* and *all*. Figure 7 shows the traffic models for the approach that takes into account IOAs in full detail and Figure 8 shows the traffic model that does not take into account information about IOAs. The model that results from combining states with overlapping IOAs is shown in Figure 9. Figure 7 shows a large number of transitions and states representing events in which various subsets of the same IOAs have been requested within the same function. We chose to present this graph in a small scale just to give an idea of the size and structure of this model. For better readability, Figure 9 has been enlarged⁴. Each state is marked with some color, and contains information on the *Direction*, *TypeID*, *ASDU address*, *IOAs* which it includes and the *count* of how many packets of this type have been observed. Each transition is labeled with its probability. Reduction *all* then merges all states with the same color as shown in Figure 8⁵.

Figures 7, 8 and 9, show that the number of states and transitions reduces considerably, when (partly) abstracting from the IOAs. The number of states in the DTMC reduces from 117 to

Table 1. Reduction gain for three different cases.

Communicating pair	Element	None	Overlapping	All	Overlapping - gain	All - gain
172.31.8.170 172.31.1.100 (example)	# states	117	17	11	6.88	10.64
	# transitions	896	51	39	17.57	22.97
172.31.10.230 172.31.1.100 (best case)	# states	189	19	10	9.95	18.9
	# transitions	1329	55	40	24.16	33.23
172.31.3.99 172.31.1.100 (worst case)	# states	11	11	9	1	1.22
	# transitions	22	22	22	1	1
average state gain					1.58	2.09
average transition gain					1.51	2.25

17 when merging the overlapping IOAs. Further reduction decreases the number of states to 10. We tested all 148 communicating pairs present in the traffic captures and calculated their respective state reduction gain. This is defined as

⁴ See also: <https://github.com/jjchromik/intravis/blob/master/example/over.pdf>

⁵ See also: <https://github.com/jjchromik/intravis/blob/master/example/all.pdf>

Fig. 8. Not taking into account IOAs

the number of states of the reference DTMC divided by the number of states of the reduced DTMC. The transitions reduction gain is defined analogously.

Table 1 provides the number of states and transitions for the original and the reduced traffic models for (i) the example shown in Figures 7, 8 and 9, (ii) the DTMC with largest reduction gain (best case), and (iii) the DTMC with smallest reduction gain (worst case). We can see that while in the best case, the reduction *overlapping* decreases the number of states almost by a factor 10, and the reduction *all* almost by a factor 19. The worst case shows almost no reduction. We also provide the average state and transition gain that we observed for all 148 communicating pairs.

4 Validation

To compare the detection rates of the reduced traffic models with respect the original one, we introduce anomalies (e.g., out-of-order packets) into the traces.⁶

For validation we use the same traffic capture mentioned in Section 3.3. This data was split into two parts of 5 days each. One half is used for training a reference DTMC using either no reduction or one of the two approaches listed in Section 3.3. The other half is used for detecting anomalies by comparing the model obtained from the remaining (testing) trace with the training traffic model. Due to the regularity of SCADA traffic the amount of data should be enough to

⁶ The code used to modify the traces is available on github <https://github.com/penc4ke/manipulateTraces.git>.

Fig.9. DTMCs representing legit activities between devices 172.31.1.100 and 172.31.8.170 over the period of 10 days. States with overlapping IOAs are combined.

Table 2. State anomalies: overall (new)

Reduction Type	No change		Copy			Remove			Swap		
			0.1%	1%	10%	0.1%	1%	10%	0.1%	1%	10%
none	11 (1)	median	11 (1)	11 (1)	19 (1)	11 (1)	11 (1)	14 (1)	11 (1)	11 (1)	20.5 (1)
		variance	0 (0)	0.233 (0)	2.456 (0)	0 (0)	0.1 (0)	0.667 (0)	0.4 (0)	0.178 (0)	0.933 (0)
overlapping	9 (1)	median	9 (1)	9 (1)	15 (1)	9 (1)	9 (1)	12 (1)	9 (1)	10 (1)	15.5 (1)
		variance	0 (0)	0.178 (0)	1.956 (0)	0 (0)	0 (0)	0.622 (0)	0.4 (0)	0.267 (0)	0.933 (0)
all	6 (0)	median	6 (0)	6 (0)	8 (0)	6 (0)	6 (0)	7 (0)	6 (0)	6 (0)	10 (0)
		variance	0 (0)	0.1 (0)	0.989 (0)	0 (0)	0 (0)	0.622 (0)	0.1 (0)	0.1 (0)	0.767 (0)

capture all relevant events. The following anomalies have been introduced: (i) copying a random packet from the used trace and adding it at a random position, (ii) removing a random packet from the trace, and (iii) swapping packets, i.e., interchanging the position of two random packets. We investigate applying the above changes to (i) 0.1%, (ii) 1%, and (iii) 10% of the packets from the testing trace. Note that while the added anomalies are not necessarily attacks, the results allow valuable insight into the accuracy of the reduced models.

4.1 Detection

We use the same algorithm and thresholds as explained in [12]. The thresholds for both a state violation and transition violation equal 0.1. To detect sequence violations we compare the differences between the trained DTMCs and testing DTMCs. In case the difference exceeds the above thresholds, an alert is raised. The detection mechanism checks the violations mentioned in Section 3.2:

New transitions violation - a transition exists in the testing DTMC, but not in the training phase. **New state violations** - a state created in the testing phase, which does not exist in the training phase. **Transition anomalies** - the transition probability in the testing DTMC differs more than the predefined threshold from the corresponding probability in the training phase.

Additionally, we provide the number of **state anomalies**, that is the number of states affected by transition anomalies.

4.2 Results and discussion

We modify the second part of the trace 10 times and compute the median and variance of the number of detected anomalies for copying, removing and swapping respectively 0.1%, 1% and 10% of all 6079 packets. Furthermore, we compare the results of the original model without reduction to the results of the two proposed reductions.

Table 2 shows the number of state anomalies, including the new states (given in brackets). The row providing the results for the original approach and the column showing the results without any changes in the trace are marked grey as they indicate the reference cases. For the non-modified trace, the original model detects 11 state anomalies, out of which 1 state was new. The new state corresponds, e.g., to a packet that occurs in the real traffic but was never seen

Table 3. Transition anomalies: overall (new)

Reduction Type	No change		Copy			Remove			Swap		
			0.1%	1%	10%	0.1%	1%	10%	0.1%	1%	10%
none	24 (10)	median	31.5 (17.5)	70 (55.5)	134 (115)	28.5 (14.5)	37 (23)	61 (46.5)	38 (24)	89.5 (75.5)	145 (127)
		variance	1.344 (1.344)	15.156 (15.211)	48.222 (36.989)	1.789 (1.789)	1.822 (1.822)	13.656 (11.333)	10.278 (10.278)	38.678 (35.333)	31.833 (32.233)
overlapping	18 (6)	median	25 (13)	57 (45.5)	98 (81.5)	22 (10)	29.5 (17.5)	49 (35)	31.5 (19.5)	71 (59.5)	104 (90.5)
		variance	2.278 (2.278)	15.956 (17.111)	34.844 (13.156)	1.433 (1.433)	0.678 (0.678)	8.278 (3.656)	8.544 (8.544)	16.233 (14.622)	12.711 (14.622)
all	14 (4)	median	17 (7)	30.5 (20.5)	49.5 (41)	15.5 (5.5)	21 (11)	33 (21)	21.5 (11.5)	37 (27)	57 (44.5)
		variance	0.267 (0.267)	8.711 (7.289)	13.733 (10.222)	0.278 (0.278)	1.333 (1.433)	2.933 (1.956)	7.122 (6.5)	15.211 (10.1)	12.622 (6.989)

in the training phase. The remaining 10 anomalous states are considered false positives, as they do not result from a modification of the original trace, but from an irregularity in the trace itself. Comparing with the two proposed reduction types, we notice that the number of false positives drops, hence, the detection accuracy of the reduced model improves, due to abstracting from the specific IOA numbers. In the reduction *all*, there are 6 anomalous states out of which none are new. This suggests that the mentioned packet uses a TypeID that appears in the training capture, but the IOA number did not.

After introducing anomalies in the traffic sequence, we can see that only copying/removing/swapping 10% of the packets increased the number of detections. The introduced anomalies are not detected as new state anomalies, since they do not introduce any new command in the trace. For anomalies that do change the number of detections, the question remains, whether the detected state anomalies are harmless or whether they indeed can harm the system.

Table 3 shows the number of transition anomalies and the number of new transition violations (provided in brackets). Again, the reference case is marked gray (reduction *none*). The original approach detects 24 transition irregularities in the original trace, out of which 10 are new. All those need to be considered as false positives. Possibly the training sequence was too short, as the dataset did not contain messages in this order. The gray column, representing the original trace shows that the reductions decrease the number of false positives w.r.t. the reference case.

Modifying the traces introduces additional transition anomalies. Even modifying 0.1 % of all packets increases the number of new anomalous transitions considerably. When reducing the traffic models, the number of detected anomalies decreases. E.g., when copying 10 % of the packets, without reduction 115 new transitions are observed, while *overlapping* results in 81.5 anomalies, and *all* in 41 anomalies. An operator may prefer fewer alerts, as too many notifications may be ignored. However, the question remains, how to distinguish an attack from a false positive alert. Note that the reduced number of detections when applying reductions stems from two sources. First, we lose false positives

as in the reference case, which increases the accuracy of detection. Second, not every change is detected in the reduced model, which decreases the sensitivity. The current detection algorithm is not performed after each event, hence we are unable to distinguish between losing false positive or true positive.

Reconsidering the disconnecter and switch attack from Section 2.3 shows that the reduction method should be chosen keeping the application in mind. If a single PLC would control the actuators, the same function (TypeId) referring to opening or closing respective IOs could appear in a specific order. Therefore, implementing reduction *all* could abstract away too much information. This could be preserved with the *overlapping* reduction, still reducing the size of the traffic model. In contrast, when dealing with simple reading commands, such as a General Interrogation, merging all IOAs would not result in a loss of accuracy, still reducing the size of the traffic model.

5 Conclusions

Commonly, SCADA traffic behaves quite regularly and results in packets sent in a predefined order. Hence, learning traffic models and comparing sequences of traffic to such models is a promising research direction. However, the developed models can easily become very large and it might not be feasible to maintain large models for each pair of communicating devices.

With this paper, we show that some cases exist where these models can be substantially reduced. In our use cases, states differing just for the range of Information Object Addresses, used in IEC-104, could be easily and conveniently combined in the DTMCs. We observe that completely abstracting from IOAs reduces the model size considerably while losing accuracy. Despite lowering down the number of false positives this may cause the IDS to overlook specific attacks, like the disconnecter and switch attack. For this reason a more conservative approach combining states with overlapping IOAs has the highest chance to succeed because of the higher model accuracy while still reducing model size. We conclude that when choosing reduction methods the actual purpose of the exchanged functions of the IEC-104 protocol should be taken into account. By understanding the goal of the actual functions (TypeIds), one can use specifically tailored reduction techniques for different functions. However, in most cases, the knowledge needed to the reduction function can come only from the operator side. Future work will focus on detecting actual attacks using a hybrid approach: either combining states with overlapping IOAs or abstracting from IOAs completely, depending on the TypeId. Moreover, the detection has to be performed in real-time, e.g., by using conformance testing techniques.

References

1. B. Zhu, A. Joseph, and S. Sastry, “A Taxonomy of Cyber Attacks on SCADA Systems,” in *Int. Conf. on Internet of Things and on Cyber, Physical and Social Computing*, pp. 380–388, IEEE CS Press, 2011.

2. G. Burke and J. Fahey, "AP Investigation: U.S. power grid vulnerable to foreign hacks," viewed 06.06.2015. <http://lasvegassun.com/news/2015/dec/21/ap-investigation-us-power-grid-vulnerable-to-forei/>.
3. D. Goodin, "First known hacker-caused power outage signals troubling escalation," viewed 06.06.2015. <http://arstechnica.com/security/2016/01/first-known-hacker-caused-power-outage-signals-troubling-escalation/>.
4. N. Falliere, L. Murchu, and E. Chien, "White paper: W32. Stuxnet dossier," tech. rep., Symantec Corp., 2011.
5. M. Caselli, E. Zambon, J. Petit, and F. Kargl, "Modeling Message Sequences For Intrusion Detection in Industrial Control Systems," *IFIP*, vol. 466, pp. 49 – 71, 2015.
6. R. R. R. Barbosa, R. Sadre, and A. Pras, "Flow whitelisting in SCADA networks," *Int. Journal of Critical Infrastructure Protection*, vol. 6, no. 3, pp. 150–158, 2013.
7. N. Goldenberg and A. Wool, "Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems," *Int. Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 63–75, 2013.
8. B. Kang, K. McLaughlin, and S. Sezer, "Towards a stateful analysis framework for smart grid network intrusion detection," in *4th Int. Symp. for ICS & SCADA Cyber Security Research 2016*, pp. 1–8, BCS Learning & Development Ltd., 2016.
9. H. Lin, A. Slagell, Z. Kalbarczyk, P. Sauer, and R. Iyer, "Runtime semantic security analysis to detect and mitigate control-related attacks in power grids," *IEEE Transactions on Smart Grid*, vol. PP, no. 99, pp. 1–16, 2016.
10. R. R. R. Barbosa, R. Sadre, and A. Pras, "A first look into scada network traffic," in *IEEE/IFIP Network Operations and Management Symposium*, pp. 518–521, IEEE CS Press, 2012.
11. C. Feng, T. Li, and D. Chana, "Multi-level anomaly detection in industrial control systems via package signatures and lstm networks," in *47th IEEE/IFIP Int. Conf. on Dependable Systems and Networks*, pp. 1–12, IEEE CS Press, 2017.
12. M. Caselli, E. Zambon, and F. Kargl, "Sequence-aware intrusion detection in industrial control systems," in *1st ACM Workshop on Cyber-Physical System Security*, pp. 13–24, ACM, 2015.
13. I. N. Fovino, A. Coletta, A. Carcano, and M. Masera, "Critical state-based filtering system for securing SCADA network protocols," *IEEE Transactions on industrial electronics*, vol. 59, no. 10, pp. 3943–3950, 2012.
14. International Electrotechnical Commission, "IEC 60870-5-104, Transmission Protocols, Network Access for IEC 60870-5-101 Using Standard Transport Profiles," 2003.
15. C. Alcaraz, J. Lopez, J. Zhou, and R. Roman, "Secure SCADA framework for the protection of energy control systems," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 12, pp. 1431–1442, 2011.
16. G. Clarke and D. Reynders, *Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems*. Newnes, 2004.
17. G. Burke and J. Fahey, "LIAN 98(en) : Protocol IEC 60870-5-104, Telegram structure," viewed 13.12.2017. http://www.mayor.de/lian98/doc.en/html/uiec104_struct.htm.
18. J. Nugteren, "ACM completes investigation into power outage in Diemen," viewed 18.12.2017. <https://www.acm.nl/en/publications/publication/16469/ACM-completes-investigation-into-power-outage-in-Diemen/>.
19. Associated Press, "Flights cancelled at schiphol airport as power outage hits amsterdam," viewed 26.06.2017. <https://www.theguardian.com/world/2015/mar/27/flights-cancelled-schiphol-airport-power-outage-amsterdam>.