ACCURATE MODELING OF THE SIEMENS S7 SCADA PROTOCOL FOR INTRUSION DETECTION AND DIGITAL FORENSICS

Amit Kleinmann, Avishai Wool
Tel-Aviv University, Tel-Aviv 69978, Israel,
amitkl@post.tau.ac.il, yash@eng.tau.ac.il

ABSTRACT

The Siemens S7 protocol is commonly used in SCADA systems for communications between a Human Machine Interface (HMI) and the Programmable Logic Controllers (PLCs). This paper presents a model-based Intrusion Detection Systems (IDS) designed for S7 networks. The approach is based on the key observation that S7 traffic to and from a specific PLC is highly periodic; as a result, each HMI-PLC channel can be modeled using its own unique Deterministic Finite Automaton (DFA). The resulting DFA-based IDS is very sensitive and is able to flag anomalies such as a message appearing out of its position in the normal sequence or a message referring to a single unexpected bit. The intrusion detection approach was evaluated on traffic from two production systems. Despite its high sensitivity, the system had a very low false positive rate - over 99.82% of the traffic was identified as normal.

Keywords: network, intrusion-detection, SCADA, S7

1. INTRODUCTION

Industrial facilities are relying to an ever-larger extent on Industrial Control Systems (ICS). The NIST Guide to ICS Security (Stouffer, Falco, & Scarfone, 2013) explains that ICS is a general term that encompasses several types of control systems, including Programmable Logic Controllers (PLC), Distributed Control Systems (DCS), Supervisory Control And Data Acquisition (SCADA) systems, and other control system configurations. An automation system within a campus is usually referred to as a DCS, while SCADA systems typically comprise of different stations distributed over large geographical areas.

1.1 Background

SCADA systems are used for monitoring and controlling numerous Industrial Control Systems (ICS). In particular, SCADA systems are used in critical infrastructure assets such as chemical plants, electric power generation, transmission and distribution systems, water distribution networks, and waste water treatment facilities. SCADA systems have a strategic significance due to the potentially serious consequences of a fault or malfunction.

SCADA systems typically incorporate sensors and actuators that are controlled by Programmable Logic Controllers (PLCs), and which are themselves managed by a Human Machine Interface (HMI). PLCs are computer-based solid-state devices that were originally designed to perform the logic functions executed by electrical hardware (relays, switches, and mechanical timer/counters). PLCs have evolved into controllers with the capability of controlling complex processes. PLCs are generally used for discrete control in discrete manufacturing.

SCADA systems were originally designed for serial communications, and were built on the premise that all the operating entities would be legitimate, properly installed, perform the intended logic and follow the protocol. Thus, many SCADA systems have almost no measures for defending against deliberate attacks. Specifically, SCADA network components do not verify the identity and permissions of other components with which they interact (i.e., no authentication and authorization mechanisms); they do not verify message content and legitimacy (i.e., no data integrity checks); and all the data sent over the network is in plaintext (i.e., no encryption to preserve confidentiality). Therefore, deploying an Intrusion Detection Systems (IDS) in a SCADA network is an important defensive measure.

The Siemens S7 is one of the leading protocols used in SCADA networks. Siemens S7 Programmable Logic Controllers (PLCs) (Siemens, 2014) are estimated to have over 30% of the worldwide PLC market (Electrical Engineering Blog, 2013). The platform is so popular that other companies (e.g., (VIPA - A Yaskawa company, 2014)) offer compatible PLCs.

1.2 Related Work

Since the S7 protocol is proprietary, there is little published information about attacks against it. An exception is the work of (Beresford, 2011). The author showed that the standard S7 protocol is not encrypted, or authenticated, it is susceptible to spoofing, session hijacking, Denial of Service (DoS) attacks, and other attacks. Gaining access to the control network gives the attacker full access to the PLCs and allows attacks against the engineering workstation as well.

(Zhu, Joseph, & Sastry, 2011) evaluated several SCADA-specific Network Intrusion Detection Systems (NIDSs), but they mentioned that, to their best knowledge, none of the surveyed systems has been tested on real operational SCADA network. Due to the lack of access to production ICS networks, many works deal with the issue of building a SCADA testbed that enables experimental capabilities of checking vulnerabilities and validating security solutions (Genge, Siaterlis, Nai Fovino, & Masera, 2012; Hahn et al., 2010; Mallouhi, Al-Nashif, Cox, Chadaga, & Hariri, 2011). In contrast, one of the important aspects of our work is that the intrusion detection approach is evaluated using real traffic from production SCADA networks.

(Yang, Usynin, & Hines, 2006) used an Auto Associative Kernel Regression (AAKR) model and applied it on a SCADA system looking for matching patterns. The AAKR model used numerous indicators, representing network traffic and hardware-operating statistics to predict the normal behavior. Hence, this model needs to monitor different indicators for different intrusion methods, and must manage a large number of potentially valuable variables.

Several recent studies (such as (Atassi, Elhajj, Chehab, & Kayssi, 2014) & (Chen, Hsiao, Yang, & Ou, 2013)) suggest anomaly-based detection for SCADA systems that is based on Markov chains. However, (Ye, Zhang, & Borror, 2004) showed that although the detection accuracy of this technique is high, the number of 'false positive' values is also high, as it is sensitive to noise.

(Hadziosmanovic, Bolzoni, Hartel, & Etalle, 2011) used the logs generated by the control application running on the HMI to detect anomalous patterns of user actions on process control application. The focus of this work was on the threats that can be triggered by a single user action. The authors acknowledged that "an attacker could manipulate logs by sending false data to the control application". This model is also susceptible to replay attacks.

(Barbosa, Sadre, & Pras, 2012) studied the periodicity characteristics of SCADA traffic. They measured dominant periods of between 1-60 seconds in their datasets. They also observed changes in the baseline patterns of the SCADA traffic they collected, which they related to the start (or end) of non periodic high throughput flows. They speculated that these changes are due to changes in the controlled environment, such as water tanks becoming full and pipes being closed.

(Cheung et al., 2007) designed a model-based intrusion detection appliance for Modbus/TCP using: (i) a protocol-level technique that verifies

Figure 1 The S7 packet encapsulated inside TCP/IP

IP Header	TCP Header	TPKT Header	COTP Header	S7 (Header and PDU)

the Modbus/TCP specifications for individual fields in Modbus/TCP messages; (ii) a communication pattern modeling technique based on Snort rules (Roesch, 1999); and (iii) a learning model that describes the expected trends in the availability of servers and services. In subsequent work, (Valdes & Cheung, 2009) incorporated adaptive statistical learning methods into the system to detect for communication patterns among hosts and traffic patterns in individual flows. More recently (Briesemeister, Cheung, Lindqvist, & Valdes, 2010) integrated the developed intrusion detection technologies into the EMERALD event correlation framework (Porras & Neumann, 1997).

(Goldenberg & Wool, 2013) developed a model-based approach (the GW model) for detecting intrusions in Modbus SCADA Networks. Following this approach, the traffic on the control network is divided into separate logical HMI-PLC channels, each containing only a single PLC's traffic. In the GW model, the HMI-PLC traffic pattern for a given channel is periodic, repeating the same sequence of queries (and matching responses) over and over. Hence each of these channels is modeled as Mealy Deterministic Finite Automaton (DFA).

One of our goals in this study is to investigate how well the GW model captures the behaviour of the S7 protocol, which is much richer than the simple Modbus protocol.

1.3 Contribution

This paper describes a model-based IDS designed for S7 networks. The detection approach is based on our observation that, like Modbus traffic, S7 traffic to and from a specific PLC is highly periodic, with the same messages being sent repeatedly according to a fixed pattern. As a result, it is possible to model each HMI-PLC channel using its own DFA. The resulting DFA-based intrusion detection system looks deep into S7 packets and produces a traffic model that captures detailed packet characteristics. Thus, the intrusion detection approach is very sensitive and is able to flag anomalies such as a message appearing out of position in the normal sequence or a message referring to a single unexpected bit.

1.4 S7 Protocol Details

Based on live S7 traffic traces collected from two production ICSs, we provide one of the first characterizations of the Siemens S7 SCADA protocol. Our starting point was the DFA based model of (Goldenberg & Wool, 2013), that was, so far, tested successfully on Modbus traffic. Using our observations from the analyzed traffic, we adapted the model to fit the richer S7 semantics. In particular, the S7 packet structure is much more complex than that of Modbus, allowing simultaneous reference to multiple variables of different data types. Moreover the grouping of variable references into packets is not always fixed, and the protocol allows multiple requests in flight. After incorporating these S7 features into the DFA-based model, our system was able to model the traffic very accurately, with extremely low false positive rates: Over 99.82% of the traffic was identified as normal.

2. THE DFA-BASED MODEL FOR MODBUS

The GW model (Goldenberg & Wool, 2013) was developed and tested on Modbus traffic. Modbus is a simple request-response protocol widely used in SCADA networks. A Modbus HMI sends a request to a Modbus PLC. The request includes a function code specifying the service, and the address range of data items. Modbus functions include reading values from coils (bit-size entities) or registers (16-bit entities), writing values to coils and registers, and performing diagnostics. After the PLC processes the request, it sends a response back to the HMI.

Figure 2 S7 0x32 PDU: Header for ROSCTR 1 or 3, Function Code 4 or 5 (Read/Write)

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Protocol Id								ROSCTR								Reserved															
Request Id																Parameter Length															
Data Length																Error Code - only for ROSCTR 3															
Function Code								Item Count																							

In the GW model, the key assumption is that traffic is *periodic*, therefore, each HMI-PLC channel is modeled by a Mealy Deterministic Finite Automaton (DFA). The DFA for Modbus has the following characteristics:

- A symbol is defined as a concatenation of the message type, function code, and address range, totaling 33-bits.
- A state is defined for each message in the periodic traffic pattern.

The GW model suggests an IDS that comprises two stages: A learning stage, where a fixed number of packets is captured, the various channels are detected, the pattern length is revealed, and a dedicated DFA is automatically built for each channel. The learning assumes that the sniffed traffic is benign; and an enforcement stage, where traffic is monitored for each channel (according to its DFA), and proper events are triggered.

The GW model categorizes input symbols into two groups: Known and Unknown. The Known group consists of all the input symbols that were observed during the learning stage, and have a matching DFA state. The Unknown symbols are all the rest. Four transition types (output symbols) are defined in the DFA: 'Normal', 'Miss', 'Retransmission', and 'Unknown'.

The key strengths of the GW model are: the fact that it goes much deeper than other intrusion detection models into the details of the SCADA protocol specification, its ability to capture inter-packet relationship, and the clear semantics of the detection process. The model was successfully evaluated against real Modbus traffic, and was able to detect real anomalies, while exhibiting an extremely low 'false positive' rate.

3. S7 COMMUNICATION OVER TCP/IP

Since the S7 protocol is proprietary and little is published about it, this section summarizes some of its key features, before delving into our modeling. The information in this section is based on the reverse engineering work of (Marsching, 2013)(Hergenhahn, 2011)(Nardella, 2014)(Wiens, 2014) augmented by our own analysis of live S7 traffic.

3.1 The S7 PLC Platform

The Siemens SIMATIC S7 product line was introduced in 1995, and includes both standard PLC models (S7-200, S7-300 and S7-400), and new generation PLCs, the S7-1200, and the S7-1500 (introduced in 2009 and 2012 respectively).

Siemens has its own HMI software for its SIMATIC products called STEP7. The Siemens S7 Ethernet Driver (Kepware Technologies, n.d.) provides connectivity to Siemens S7 devices via the Siemens TCP/IP Ethernet protocol.

The memory of an S7 PLC is divided into different areas (see (Kepware Technologies, n.d.)). Of these, the Data Blocks area is used to store the internal state of the program running on the PLC. Within the Data Blocks area, each Data Block is identified by a 16-bit DB number, and contains multiple data items each with a 24-bit address. Thus a location in the PLC memory is identified as follows:

- For the Input, Output, Peripheral, and Marker Flags memory areas - by 32 bits consisting of its Area code, and Address.
- For the Data Block memory area - by 48 bits consisting of its Area code, DB number, and Address.

S7 has a rich set of data types. Each variable, stored at a given location, has a type such as: Bit, Byte, Int, Real, etc., which implies a length (Transport size), see (Kepware Technologies, n.d.).

Apart from the Siemens S7 Ethernet driver (Kepware Technologies, n.d.), there are other 3rd-party communication suites for interfacing and exchanging data with Siemens S7 PLCs (e.g., The LIBNODAVE library (Hergenhahn, 2011), and Snap7 (Nardella, 2014)). In addition there is now a Wireshark dissector for S7 communication (Wiens, 2014). These software libraries have been developed through reverse-engineering.

The TCP/IP implementation of the S7 protocol relies on ITOT (Rose & Cass, 1987) and communicates across the well known TCP port 102. S7 works on top of the ISO Connection Oriented Transport Protocol (COTP) (ISO 8073:1986(E), 1986)(McKenzie, 1984) and TPKT (Rose & Cass, 1987). Both TPKT and COTP add their own headers (inside the TCP segment). Thus the S7 packet is encapsulated within the COTP packet, see Figure 1.

The S7 protocol defines formats for exchanging S7 messages between devices. A device can take the role of a client, a server or a peer. In a client-server communication mode, the HMI (client) device initiates the transactions (called queries) and the PLC (server) responds by supplying the requested data to the client, or by taking the action (e.g., write operation) requested in the query. In a peer-to-peer communication mode, partner (peer) devices can exchange unsolicited data, i.e., once the connection is established, both can send data to the other partner. In the data we collected, only client-server communication was observed.

Two different protocol flavours are implemented by SIMATIC S7 products: The standard SIMATIC S7 PLCs implement an S7 flavor that is identified by the protocol number 0x32, while the new generation PLCs implement an S7 flavor that is identified by the protocol number 0x72. Among other changes, the newer S7-0x72 protocol also supports security features. The two flavors are quite different, and in fact the Wireshark dissector (Wiens, 2014) only supports the 0x32 flavor. In this article we restrict our investigation to the S7-0x32 protocol.

3.2 The S7-0x32 PDU

The S7 PDU size is bounded. Its maximum length varies between 112 and 960 bytes (see (Siemens, 2013)), and is negotiated during the connection.

A single S7 PDU can reference multiple ranges of PLC variables. The PDU is divided into three parts: a fixed header, a parameters part and a data part. In a request message the parameters part indicates which PLC variables are being accessed, and the optional data part includes the values that need to be written into the variables (in a write command). The data part includes a list of the write values (in case of write request) or a list of read results (in case of read response).

The PDU header includes nine fields (see Figure 2), of which the following are important to our work:

- The Protocol Id is constant and is set to 0x32.
- The ROSCTR (Remote Operating Service Control) field can take the values: 0x01 (request), 0x02 (acknowledge), 0x03 (response), 0x07 (request user data), or 0x08 (request server control). We only observed request and response in our data.
- The Request Id field is used to synchronize between a request packet and a response packet that are in flight (see Section 5.3).
- The Parameter Length, and Data Length fields specify the length (in bytes) of the parameter part and the data part of the PDU.
- The Function Code field encodes operations such as communication setup, system info, data read/write, block move, and PLC control functions.
- The Item Count field (only in data read/write functions).

Figure 3 S7 0x32 PDU: Read/Write Request - Parameter Item

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Var Type								Var Length								Syntax Id								Transport Size							
Length																DB Number															
Area								Address																							

Figure 4 S7 0x32 PDU: Header of Data Item

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Return Code								Transport Size								Data Length (per transport size)															

Following the PDU header, there is a parameter area that consists of Item Count parameter items (only in read/write request packets). Each parameter item includes (see Figure 3):

- The variable specification, which includes three fields (Type, Length and Syntax Id) that imply certain attribute settings of the variables in this item (e.g., permitted address range, supported data types and access permissions).
- A Transport Size field encodes the type (and thus the length) of each variable/s of the parameter item (see (Kepware Technologies, n.d.)).
- A Length field which specifies the number of referenced variables (note that a single parameter item can refer to a range of variables).
- An Area field which is an 8-bit identifier of the PLC's memory area.
- A DB (Data Block) Number field which is a 16-bit identifier of a Data Block within the Data Blocks area.
- Address field which is a 24-bit address of a data item.

The number of parameter items within a PDU is limited to 255. However both the LIBNODAVE library (Hergenhahn, 2011) and the open source code of Snap7 (Nardella, 2014) enforce a maximum of 20 parameter items per PDU. In the data we captured all the packets included at most 19 items per PDU, and in fact most of the PDUs of trace #2 and trace #3 included exactly 19 items.

Following the parameter area, there is a list of data items (only in write request packets, or read response packets). Data items include the data header and the data itself. The structure of a data header is depicted in Figure 4.

4. COLLECTED DATASETS

(Garitano, Uribeetxeberria, & Zurutuza, 2011) emphasize that conducting SCADA IDS research based on traffic simulation has several risks, such as lacking realism "that affects everyday use of SCADA systems". In order to test a proposed IDS approach as realistically as possible, it is imperative to use real SCADA traffic.

In this study we analyze three different traces that were collected at real ICS facilities. The first S7 SCADA trace was collected at a manufacturing plant in the building material industry. The ICS here controls a manufacturing process where raw materials are automatically weighed and measured in a mixer. In this trace we observed a single channel between the HMI and an S7-compliant VIPA PLC. The next two traces were collected at a waste water treatment facility. The SCADA system in this plant controls the maintenance of specific levels in tanks, flow rates, and temperatures and pressures at certain processes. It can also execute logic for automating the starting and stopping of pumps, opening and closing of valves, and other control functions. In these two traces we observed

Table 1 Overview of the datasets

#	Description	Duration	TCP Packets	S7 Packets
1	Building material manufacturing plant	3242 Sec.	56843	35747
2	Waste water treatment facility	1691 Sec.	38962	21670
3	Waste water treatment facility	1204 Sec.	42465	25379

Table 2 Results of applying the basic model on S7 traffic

Dataset	AER	Pattern	# Normal	# Unknown	# Miss	% Normal	% Unknown	% Miss
#1 Part 1	8.01	8	6900	30	0	99.57	0.43	0.00
#1 Part 2	11.99	12	11592	42	0	99.64	0.36	0.00
#1 Part 3	12.09	12	17076	86	0	99.57	0.43	0.00
#2 All	10.96	12	18087	374	123	97.33	2.01	0.66
#3 Part 1	10.81	12	1392	11	3	99.00	0.78	0.21
#3 Part 2	20.13	22	20488	646	99	96.49	3.04	0.47

three channels: there were two Siemens S7-200 PLCs (each using a SIMATIC ET 200S I/O Module), and for one of them, the HMI used two TCP sockets simultaneously. An overview of the dataset properties can be found in Table 1.

The traces were collected using a Wireshark program we installed on the Microsoft Windows 7 PC running the HMI. The traces, structured in a PCAP format, were later analyzed using both Wireshark and our own software.

5. BUILDING A DFA-BASED MODEL FOR S7 TRAFFIC

In the GW model (Goldenberg & Wool, 2013) a channel is identified by the IP addresses of the HMI and the PLC, plus the Modbus unit id within the PLC. The S7 protocol does not have a unit id concept, however we did observe that the HMI sometimes opens multiple concurrent TCP connections to the same PLC. Thus we opted to identify a channel by the IP addresses of the HMI and PLC, plus the TCP source port of the HMI (the TCP destination port of the PLC is always 102).

5.1 A Basic Model of S7

Following (Goldenberg & Wool, 2013), in order to construct the model's DFA, we need to define the symbols, i.e., we need to select which S7 fields constitute a symbol in the DFA's alphabet. In our basic model we selected the following fields of the S7-0x32 PDU: ROSCTR, Parameter Length, Data Length, Error Code (if exists), Function Code, Item Count, all the parameter items (if they exist), and all the data header fields of data items (if they exist). We chose not to include the 'Request Id' field, since it does not encode any information related to data acquisition or control (it is only used to synchronize request and response packets, see section 5.3). As in (Goldenberg & Wool, 2013), we did not include the data part of the data items, since the inherent variability in the data items makes it unsuitable to be modeled by a reasonably sized alphabet. We defer the analysis of the data part to future research.

Recall that an S7 packet can refer to multiple item ranges, each of which is identified by a (32-bit or 48-bit) address location, as well as an individual type and length. Assuming (as in (Hergenhahn, 2011)&(Nardella, 2014)) a maximum of 20 parameter items per S7 PDU, the raw number of bits in a symbol can vary between 88-1992 bits. For practical reasons, we decided to fix the length of the symbol at 64 bits. Therefore, we use the SHA-1 hash function to digest the concatenation of the fields, and take the 64 least significant bits as the symbol.

In our analysis we divide time into 5-second 'time frames'. We also define the *Average Event Rate (AER)* as the mean number of symbols per second. We further define an *S7 quiescent period* as a time frame during which only 'Normal' transitions are observed.

We ran the model's learning stage on the collected datasets with a maximum pattern length of 75 symbols and a validation window of 300 $(75 \cdot 4)$ symbols. Then we ran the enforcement stage on the full datasets using the learned patterns. Table 2 shows the basic results, using the above definitions, and Figure 5 shows the detection of 'Unknown' symbols, after applying the model on the various traces. The division of trace #1 and trace #3 into parts is discussed in section 6.1.

As observed in (Goldenberg & Wool, 2013) on Modbus, Table 2 shows that the DFA accurately models the vast majority of the S7 packets. Even in this basic DFA model, over 96.49% of all packets are identified as 'Normal' packets. Over 98.35% of the time frames are S7 quiescent periods. Note that the learned pattern lengths are nearly identical to the AER, indicating that the systems had inherent 1-second periods.

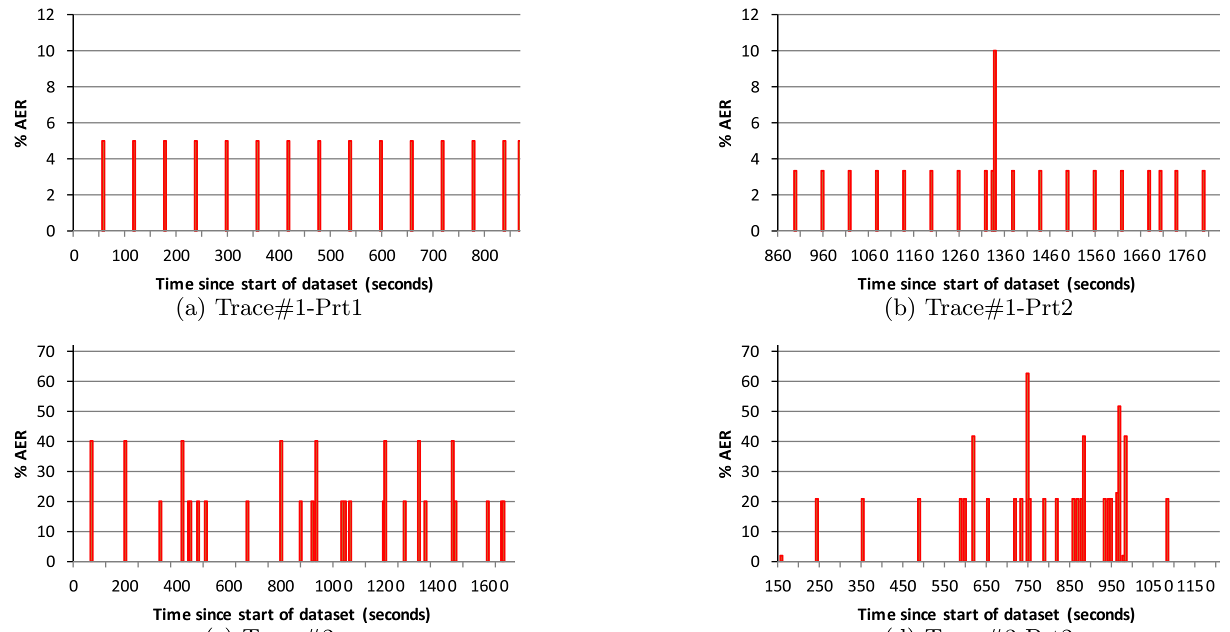
While we observe a relatively low amount of 'Unknown' symbols and almost no 'Miss' symbols in trace #1, Table 2 shows about 2-3% 'Unknown' symbols and numerous 'Miss' symbols in trace #2 and in the main phase (part 2) of trace #3 (see also Figures 5c and 5d), which indicate significant anomalies. In addition trace #2 and trace #3 exhibit substantial 'Miss' rates (see Figures 6a and 6b). The next sections analyze the cause for these anomaly events, and describe the modeling steps we took to mitigate them.

5.2 Splitting an S7 Packet into Separate Items

After inspecting many 'Unknown' symbols in trace #2 and trace #3, we made the following observation: in these traces there are changes in the grouping of S7 items within the S7 PDU. During these changes, the order of items within the stream is kept constant, but some of the items are grouped differently into PDUs. E.g., in trace #2 the pattern has 12 symbols (6 requests and 6 responses) accessing 99 items in total. Occasionally, 18 of the items that typically appear in the 2nd request PDU are shifted to the 3rd request "pushing" other items into later requests. Each of these grouping changes is reflected in the basic model by one or more 'Unknown' symbols (and 'Unknown' events). We assume that these grouping changes do not indicate an anomaly, but rather a benign variation in the acquisition timing or in the S7 transport service.

Following the above observation, we decided to avoid these 'false alarms', by applying the model at a finer granularity. Out of each original S7 packet that includes N items, we created an artificial packet per item (a total of N artificial packets per original packet). Beside the original S7 item, we copied into the artificial packet all the fields of the original packet, that do not include information related to the grouping of items within the PDU. For example, the ROSCTR field was copied, but the parameter length field, and the data length field, were ignored. We then applied the model to the artificial packets. Since each S7 original packet may include up to 20 parameter items, we set the maximum pattern length to 1500 $(75 \cdot 20)$ symbols and the validation window to 6000 $(1500 \cdot 4)$ symbols.

Table 3 shows the results after applying the model with PDU-splitting on S7. The table shows that all the 'Unknown' symbols were eliminated from trace #2, and the 'Unknown' rate in trace #3 dropped from 3.07% to only 0.07%. Further, the 'Miss' rate in these two traces improved significantly as well, from 0.66% and 0.47% in Table 2 down to 0.18% and 0.24% after PDU splitting. Before PDU-splitting, Figures 6a and 6b show 'Miss' spikes up to 13% AER and 8% AER respectively. After PDU-splitting, Figures 6c and 6d show that the peak 'Miss' spikes are under 5% AER.



(a) Trace#1-Prt1
(b) Trace#1-Prt2
(c) Trace#2
(d) Trace#3-Prt2

Figure 5 Detected 'Unknown' symbols after applying the model on the datasets. Each time frame on the X axis represents 5 seconds. The Y axis shows the events as the percentage of the Average Event Rate (AER) for each time period.

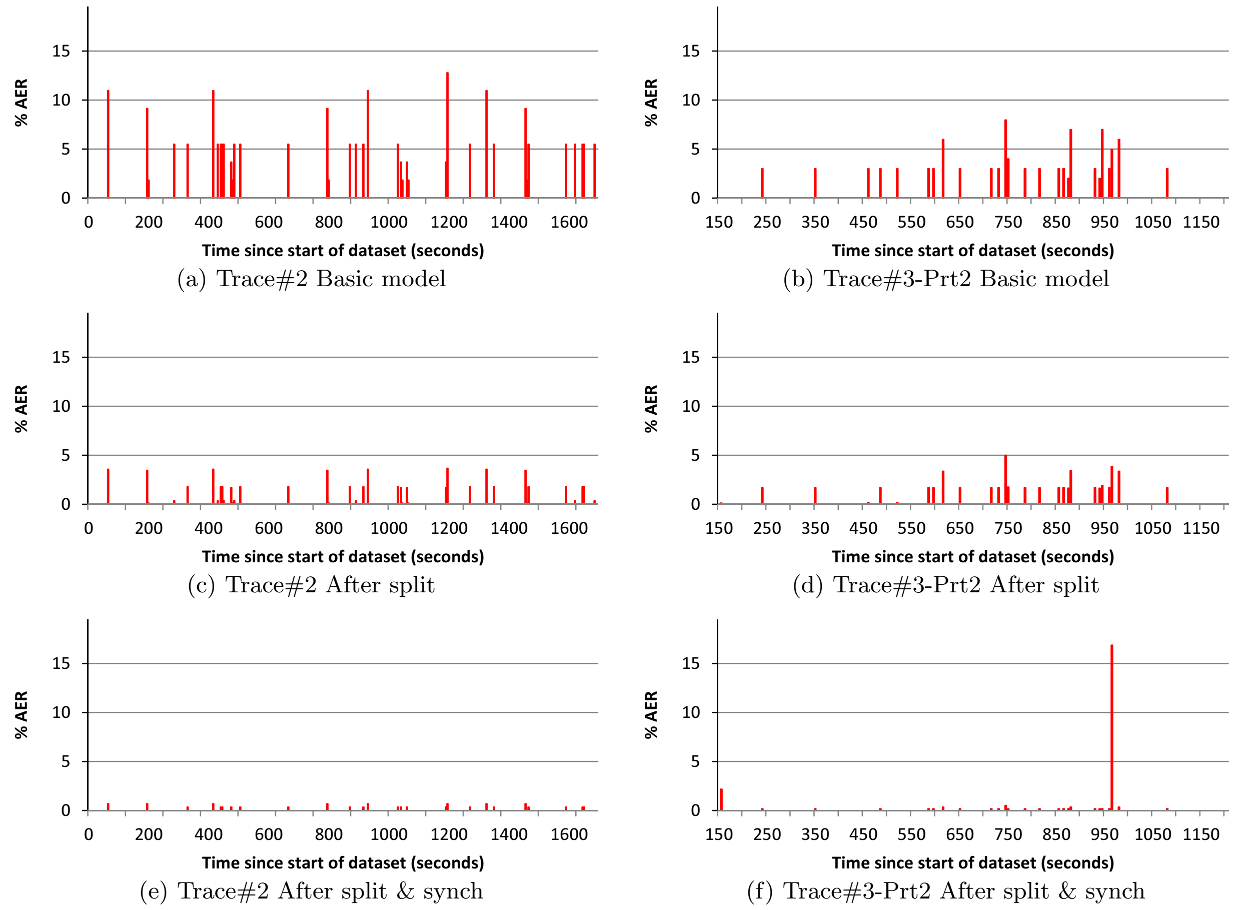
Table 3 Results after applying the model with PDU-splitting on S7 traffic. Note that the Average Event Rate (AER) is much higher in comparison to Table 2 since each raw event (PDU) is replaced by multiple artificial 1-item PDUs.

Dataset	AER	Pattern	# Normal	# Unknown	# Miss	% Normal	% Unknown	% Miss
#1 Part 1	45.89	46	39667	28	1	99.92	0.07	0.01
#1 Part 2	63.78	64	61824	42	0	99.93	0.07	0.00
#1 Part 3	58.18	58	82535	86	0	99.90	0.10	0.00
#2 All	180.24	198	304949	0	565	99.82	0.00	0.18
#3 Part 1	178.19	198	23150	0	16	99.93	0.00	0.07
#3 Part 2	370.20	406	389335	272	950	99.69	0.07	0.24

5.3 Synchronizing In-Flight Request and Response Packets

In (Goldenberg & Wool, 2013) the model implicitly assumes that in a clean capture of benign traffic, each query packet is succeeded by its corresponding response packet. We observed that this assumption holds for trace #1. However in the S7 streams recorded in trace #2 and trace #3, we noticed that a request is not always immediately followed by its corresponding response. Instead the HMI sometimes issues several requests before receiving the corresponding responses. As a result, there might be several requests that are simultaneously in flight on the same S7 channel. The number of packets in flight and their location within the pattern varies, and is apparently influenced by timing variations. These variations in packet order are reflected in the DFA model by 'Miss' events.

In this case as well, we assume that this phenomenon is benign. To avoid false alarms we decided to synchronize each request with its associated response, before taking the DFA step, and to verify that a request packet and its corresponding response packet have the same Re-



(a) Trace#2 Basic model

(b) Trace#3-Prt2 Basic model

(c) Trace#2 After split

(d) Trace#3-Prt2 After split

(e) Trace#2 After split & synch

(f) Trace#3-Prt2 After split & synch

Figure 6 Detected 'Miss' symbols before and after applying the model with splitting and synchronization (on trace #2 and trace #3). Each time frame on the X axis represents 5 seconds.

quest Id value. The synchronization is applied during both the DFA learning stage and the DFA enforcement stage. We achieve this synchronization by queueing the request packets, and handling each request packet only when its corresponding response packet is captured. Specifically, when a request packet is detected, we first check whether the associated symbol is known. In case it is known the packet is appended to the queue. Otherwise, a DFA step is taken causing an 'Unknown' event, thus providing an immediate detection of 'Unknown' symbols. When a response is detected, a request is taken out of the queue and a DFA step is taken. Then the response is taken as an input to another DFA step.

Packets that cause 'Miss' events deserve special treatment. If left untreated, requests whose responses are missed could cause the queue to grow indefinitely. Therefore, if the DFA step of the response results in 'Miss' event, we take all the requests whose corresponding responses were missed, out of the queue.

Table 4 shows the results after applying the model with both splitting and synchronization of requests and responses (using a queue) on S7 traffic. The detected 'Miss' symbols are depicted in Figures: 6e and 6f. Trace #2 now features 5.5 times fewer 'Miss' symbols and in trace #3 we detect less than half of the 'Miss' symbols (compared to the detection after applying the model with splitting only).

6. CHALLENGES FOR THE DFA-BASED MODEL

In general, we observed that the DFA-based model is very effective on the S7 data we collected. Table 4 shows that using PDU-splitting

Table 4 Results after applying the DFA with both PDU-splitting and synchronization

Dataset	AER	Pattern	# Normal	# Unknown	# Miss	% Normal	% Unknown	% Miss
#2 All	180.24	198	305412	0	102	99.97	0.00	0.03
#3 Part 1	178.19	198	23163	0	3	99.99	0.00	0.01
#3 Part 2	370.20	406	389830	272	436	99.82	0.07	0.11

and synchronization over 98.82% of all traffic was identified as 'Normal', i.e., the model has very low 'false positive' rate over benign traffic. Nonetheless we did observe two phenomena that challenge the model and require additional research.

6.1 Multiple Phases with Irregular Periods Between Them

While in trace #2 we observed a homogeneous periodic pattern, in trace #1 and trace #3 we discovered multiple periods each with its own distinct pattern. We call these distinct periods *phases*. A phase transition is a change in the control system that results in a transition from one phase to another phase. Using the DFA learned in one phase to enforce the model during another phase results in very high anomaly rates.

To automatically identify the phases, we used the following method. In order to detect the start of a phase, we define a *Learn Threshold (LT)*. If the performance of the model learning stage is below LT, the learning is considered unsuccessful. In that case the recording and a new learning stage are restarted K symbols ahead of the start symbol of the failed stage. As soon as the learning stage succeeds, a beginning of a new phase is marked.

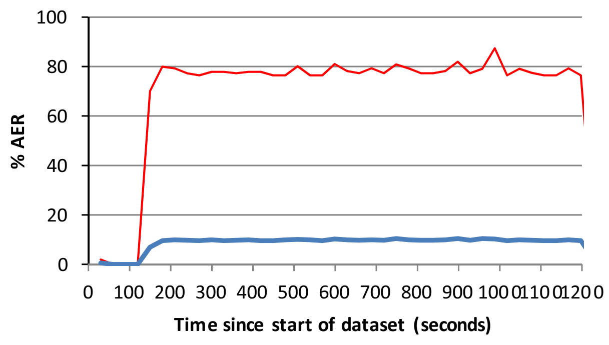
In order to detect the end of a phase, we define several threshold metrics and an algorithm. We define a *Noise Miss Threshold (NMT)* and a *Noise Unknown Threshold (NUT)* to be the rate (in terms of % AER) of 'Miss' and 'Unknown' symbols respectively, in a specific time frame. If the rate of 'Miss' symbols is more than 'NMT' or the rate of 'Unknown' symbols is more than 'NUT', the time frame is considered noisy. We define a *Phase-End Check(PEC)* Window as a sliding window of S consecutive time frames, during which we check for phase end. The *Phase End Threshold (PET)* is defined as the number of noisy time frames in the PEC Window that triggers the detection of a phase end.

We used this method on our datasets with time period = 0.5 seconds, with NMT = NUT = 80% (i.e., a time frame with over 80% 'Miss' or 'Unknown' transitions is considered Noisy), with PEC window of S = 8, and with PET = 3 Noisy time frames indicating a phase end.

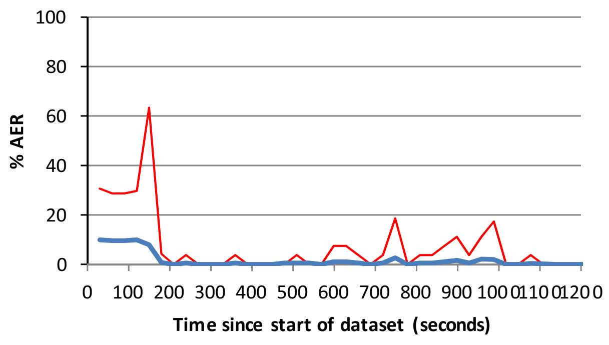
Figure 7 demonstrates the two phases of trace #3. Figure 7a shows the detected anomalies using a DFA that was learned at the beginning of the trace. Figure 7b shows the detected anomalies over the same trace with a DFA that was learned at the beginning of the 2nd phase (starting at time 155 sec.). Notice how each DFA models its own phase well, but does poorly in the other phase.

In our data, these phase changes correspond to human operator actions. Approximately 2-3 minutes after initiating the data capture, the operator adjusted some settings in the 'sewage treatment process' HMI. We believe that this action caused a brief transitional phase in the traffic, after which the HMI-PLC channel settled into a new communication pattern, which is modeled as the second phase. Barbosa et al. (Barbosa et al., 2012) noticed similar phenomena and related them to momentary increase/decrease in the amount of variables requested by a monitor and/or in the rate in which the variables are requested.

We see that detecting the start and end of a phase is fairly straight forward. However, clearly each phase requires its own learning and its own DFA. Learning and enforcing multiple DFAs is a challenge for the approach, which we leave for future work.



(a) Trace#3-Prt1



(b) Trace#3-Prt2

Figure 7 Multiple phases in the trace. Each time frame on the X axis represents 30 seconds. The thin upper line indicates 'Unknowns', the thick lower line indicates 'Miss' transitions.

6.2 Simultaneous Slower Cycles

As shown in Table 3, trace #1 has a very low 'Unknown' rate (0.07%). However, unlike in trace #3, the 'Unknown' symbols are not caused by isolated operator actions.

All three phases of trace #1 have a periodic pattern of 'read' requests and responses with a period of $T_1 = 1$ sec. However, in this trace we observed a second periodic pattern of 'write' requests and responses that is much slower, with a period of T_2=60 seconds. This periodic pattern is visible by the repeating spikes of 'Unknown' symbols depicted in Figures 5a and 5b. A similar phenomenon was observed by (Goldenberg & Wool, 2013); the captured Modbus traffic had 3 periods: a 1-sec period, a 15-minute period, and a 24-hour period. (Goldenberg & Wool, 2013) dealt with multi-period patterns using a multi-level DFA. We have not yet addressed the issue of multi-period traffic in S7. Having long periods does pose a challenge for a DFA-based IDS, since learning and enforcing long periods (with thousands of symbols) is cumbersome and error prone.

7. CONCLUSIONS AND FUTURE WORK

In this paper we developed and applied a DFA-based IDS to S7 traffic. Based on detailed analysis of captured traffic from two production ICS plants we were able to reverse-engineer the key semantics of the proprietary S7 protocol. We then modified the earlier DFA model that was developed for the simple Modbus protocol to make it suitable to the rich data model and protocol semantics of S7.

Evaluating the resulting model on our captured traces we saw that, as in Modbus, the DFA-based approach is very successful in modeling benign S7 data with over 99.82% accuracy and extremely low false positive rates. Further, the IDS is extremly efficient: it keeps minimal state during the enforcement stage, and can easily work at line-speed for real-time anomaly detection.

We have also observed two phenomena that challenge the approach and require further research: operator actions that cause short term un-modeled events and long term phase transitions, and channels with multiple (slow) periods. Additional research is also required to evaluate the approach's effectiveness against malicious traffic of various types.

Finally, note that the S7 PDU includes multiple variable-length fields, with complex encoded lengths. This structure offers many opportunities for buffer overflows in both the PLC and HMI code. We did not attempt any fuzzing against the devices in this work.

REFERENCES

Atassi, A., Elhajj, I. H., Chehab, A., & Kayssi, A. (2014). *The state of the art in intru-*

sion prevention and detection. Auerbach Publications.

Barbosa, R., Sadre, R., & Pras, A. (2012, April). A first look into SCADA network traffic. In *Ieee network operations and management symposium (NOMS)* (p. 518-521).

Beresford, D. (2011, July). Exploiting Siemens Simatic S7 PLCs. In *Black Hat USA*.

Briesemeister, L., Cheung, S., Lindqvist, U., & Valdes, A. (2010). Detection, correlation, and visualization of attacks against critical infrastructure systems. In *Eighth annual international conference on privacy security and trust (pst)* (pp. 17–19).

Chen, C.-M., Hsiao, H.-W., Yang, P.-Y., & Ou, Y.-H. (2013, Aug). Defending malicious attacks in cyber physical systems. In *IEEE 1st International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA)* (p. 13-18).

Cheung, S., Dutertre, B., Fong, M., Lindqvist, U., Skinner, K., & Valdes, A. (2007). Using model-based intrusion detection for SCADA networks. In *Proceedings of the SCADA security scientific symposium* (pp. 127–134).

Electrical Engineering Blog. (2013, May). *The top most used PLC systems around the world.* Electrical installation & energy efficiency. (Available at: `http://engineering.electrical-equipment.org/electrical-distribution/the-top-most-used-plc-systems-around-the-world.html`)

Garitano, I., Uribeetxeberria, R., & Zurutuza, U. (2011). A review of SCADA anomaly detection systems. In E. Corchado, V. Snasel, J. Sedano, A. Hassanien, J. L. Calvo, & D. Slezak (Eds.), *Soft computing models in industrial and environmental applications, 6th international conference soco 2011* (Vol. 87, p. 357-366). Springer Berlin Heidelberg.

Genge, B., Siaterlis, C., Nai Fovino, I., & Masera, M. (2012). A cyber-physical experimentation environment for the security analysis of networked industrial control systems. *Computers & Electrical Engineering.*

Goldenberg, N., & Wool, A. (2013, June). Accurate modeling of modbus/tcp for intrusion detection in scada systems. *International Journal of Critical Infrastructure Protection*, *6*(2), 63–75.

Hadziosmanovic, D., Bolzoni, D., Hartel, P. H., & Etalle, S. (2011, September). MELISSA: Towards automated detection of undesirable user actions in critical infrastructures. In *Proceedings of the european conference on computer network defense, ec2nd 2011, gothenburg, sweden* (pp. 41–48). USA: IEEE Computer Society. http://eprints.eemcs.utwente.nl/20502/.

Hahn, A., Kregel, B., Govindarasu, M., Fitzpatrick, J., Adnan, R., Sridhar, S., & Higdon, M. (2010). Development of the PowerCyber SCADA security testbed. In *Proceedings of the sixth annual workshop on cyber security and information intelligence research* (p. 21).

Hergenhahn, T. (2011, February). *LIBNODAVE, exchange data with Siemens PLCs.* (Available at: `http://libnodave.sourceforge.net`)

ISO 8073: Information processing systems – open systems interconnection – connection oriented transport protocol specification (Vol. 1986) [Standard]. (1986). Geneva, CH.

Kepware Technologies. (n.d.). *Siemens TCP/IP Ethernet - driver help.* (Available at: `http://www.kepware.com/Support_Center/SupportDocuments/Help/siemens_tcpip_ethernet.pdf`)

Mallouhi, M., Al-Nashif, Y., Cox, D., Chadaga, T., & Hariri, S. (2011). A testbed for analyzing security of SCADA control systems (tasscs). In *IEEE innovative smart grid technologies (isgt)* (pp. 1–7).

Marsching, S. (2013, October). A new EPICS device support for S7 PLCs. In *Proceedings of the 14th international conference on accelerator & large experimental physics control systems (icalepcs2013).*

San Francisco, CA, USA.

McKenzie, A. M. (1984, April 1). *RFC 905: ISO transport protocol specification ISO DP 8073.*

Nardella, D. (2014, January). *Snap7 1.2.0 - reference manual - rev. 3.* (Available at: `http://snap7.sourceforge.net`)

Porras, P. A., & Neumann, P. G. (1997, oct). EMERALD: event monitoring enabling responses to anomalous live disturbances. In *1997 national information systems security conference.*

Roesch, M. (1999). Snort - lightweight intrusion detection for networks. In *Proceedings of the 13th usenix conference on system administration* (pp. 229–238). Berkeley, CA, USA: USENIX Association.

Rose, M. T., & Cass, D. E. (1987, May 1). *RFC 1006: ISO transport services on top of the TCP: Version 3.*

Siemens. (2013, August). *SIMATIC NET, configuration limits for products of the SIMATIC NET PC software v12, application manual.* (Available at: `http://support.automation.siemens.com/dnl/jg/jgxNDg2NQAA_15227599_FAQ/15227599_QuantityStructure_and_PerformanceData_V12_e.pdf`)

Siemens. (2014). *Modular PLC controllers SIMATIC S7.* (Available at: `http://www.automation.siemens.com/mcms/programmable-logic-controller/en/simatic-s7-controller`)

Stouffer, K. A., Falco, J. A., & Scarfone, K. A. (2013, May). *Guide to industrial control systems (ICS) security* (Tech. Rep. No. 800-82). Gaithersburg, MD: National Institute of Standards and Technology (NIST).

Valdes, A., & Cheung, S. (2009). Communication pattern anomaly detection in process control systems. In *Ieee conference on technologies for homeland security (hst)* (pp. 22–29).

VIPA - A Yaskawa company. (2014). *VIPA control systems.* (Available at: `http://www.vipa.com/en/products/control-systems`)

Wiens, T. (2014, January). *S7comm wireshark dissector plugin.* (Available at: `http://sourceforge.net/projects/s7commwireshark`)

Yang, D., Usynin, A., & Hines, J. (2006). Anomaly-based intrusion detection for SCADA systems. In *5th intl. topical meeting on nuclear plant instrumentation, control and human machine interface technologies (npic&hmit 05)* (pp. 12–16).

Ye, N., Zhang, Y., & Borror, C. (2004, March). Robustness of the markov-chain model for cyber-attack detection. *IEEE Transactions on Reliability*, *53*(1), 116-123.

Zhu, B., Joseph, A., & Sastry, S. (2011, Oct). A taxonomy of cyber attacks on SCADA systems. In *Internet of things (iThings/CPSCom), 2011 international conference on and 4th international conference on cyber, physical and social computing* (p. 380-388).