Whitelisting for Characterizing and Monitoring Process Control Communication

Andreas Paul[1](✉), Franka Schuster[2], and Hartmut König[2]

[1] Codewerk GmbH, Karlsruhe, Germany
andreas.paul@codewerk.de

[2] Brandenburg University of Technology Cottbus-Senftenberg, Cottbus, Germany
{franka.schuster,hartmut.koenig}@b-tu.de

Abstract. In recent years, industrial control systems (ICS) used in critical infrastructures have come under the spotlight as a powerful target to potentially harm broader segments of society. Although there is a growing body of anomaly detection approaches in this field, the homogeneous network traffic narrative that is supposed to justify their potential success is poorly proven. At the same time, more and more machine learning (ML) schemes have been developed for this purpose neglecting though that ML is not the ideal approach for various profound detection aspects in operational technology (OT) networks. In this paper, we present and evaluate a communication whitelisting approach for anomaly detection in OT networks and point out advantages of this allegedly ancient monitoring method compared to machine learning. For this, we introduce measures to express the variability of network traffic and use them to quantify the communication dynamics of traffic for different OT infrastructures and network layers. We show that due to the static network communication in the OT domain the detection capability is sufficiently high without whitelist explosion or runtime concerns.

Keywords: Intrusion Detection · Industrial Networks · Whitelisting · Traffic Ananlysis

1 Introduction

In recent years, information technology (IT) and operational technology (OT) have converged very closely. This trend has serious implications for the protection of critical infrastructures from external and internal threats. In contrast to office networks, the traffic in OT environments is commonly expected to be static and homogeneous regarding the involved devices and their communication characteristics. Hence, many approaches assume regular network communication for the training phase to apply *anomaly detection* for the analysis. This reflects today's attack trends, which are increasingly characterised by polymorphic aspects, but at the same time must address the need to identify unknown (zero-day) attacks, which is not possible with the complement misuse detection approach.

S. Li et al. (Eds.): NSS 2023, LNCS 13983, pp. 23–45, 2023.
https://doi.org/10.1007/978-3-031-39828-5_2

For this, machine learning (ML) has meanwhile become the dominant approach because it is considered to be more efficient for the definition of normality than rule-based approaches. From the domain perspective, however, network-based attack detection relies in no small part on communication aspects that ML cannot model effectively. This comprises the precise characterization of *communication relations* which includes both the devices involved and the protocols and message types used for data transmission. Here, the strength of ML to generalise is simultaneously its weakness. Since this capability arises from abstractions of the training data, it means that some important aspects are not reflected accurately enough in the model. This leads to incorrect classifications for aspects that can be expressed with rules. Especially in OT networks, which are less dynamic than IT networks, it is an essential requirement to be able to precisely and completely recognise the devices seen in the training phase and their communication relations[1] and without any generalization[2] for the detection. Consequently, the question is not whether to apply machine learning *or* rules for attack detection. It is actually, which method is the right one for which aspects. While rules allow a precise monitoring of well-defined communication behavior, machine learning is supposed to be applied to traffic parts for which it is not reasonable or feasible to express them by rules, e. g., the application payload of packets spanning a comparably much broader space of information which is in general challenging to model. Apart from precise detection, communication rules can effectively support a key problem in OT network security research: they enable the analysis of characteristics and dynamics of OT network traffic to prove or disprove common assumptions of OT network communication compared to IT traffic. The following essential contributions of this paper are derived from this:

- We present an approach for the automated generation of communication rules from raw network traffic usable as whitelists in anomaly detection for a wide range of OT protocols independent of their underlying protocol stack.
- We apply the rule generation on six datasets taken from *real* OT networks and compare them to traffic from an OT [8] and an IT [22] testbed that are prominent in research. By quantifying the emergence of the rules over time, we measure the dynamics in network communication and express the static nature of OT network traffic that are commonly expected in the literature, but to the best of our knowledge have not been demonstrated so clearly, yet.
- We show that different OT network layers can be identified by a statistical clustering applied on the metrics derived from the rule generation process that allows us in turn to identify network layers from an unknown domain.
- We evaluate the efficiency of the rules regarding detection capability and runtime. The former is done through attacks in a public OT dataset prominent in research [8], the latter by converting the rules into a rule set for a widely

[1] A communication observation $dev_A \rightarrow dev_B$ and $dev_B \rightarrow dev_C$ may not be part of the abstraction without proper reflection in the model when using ML.

[2] The same two observations could lead to a model reflecting also $dev_A \rightarrow dev_C$ when applying ML.

used Intrusion Detection System (IDS) [21] and their application on real networks.
- In the course of the discussion regarding the attack detection capability of our rules we precisely explain the impact of the attacks contained in the used public OT data set on the network traffic. We conclude that in turn our whitelisting can be used both for packet-based labeling of anomalous data and as a baseline for more advanced detection mechanisms.

The remainder of this paper is organised as follows. In Sect. 2, we give background information of OT infrastructures and preliminary considerations for communication whitelisting. The method for whitelist generation is explained in Sect. 3. After introducing the datasets used for our evaluation in Sect. 4, the approach and the results for measuring communication dynamics are presented in Sect. 5. Practical results of the whitelist application are presented in Sect. 6. We discuss related work in Sect. 7 before concluding this paper with some final remarks in Sect. 8.

2 Preliminaries

Before presenting our approach we dive into necessary background and terminology for assessing the concepts of the paper. There are two types of industrial control systems (ICSs): distributed control systems (DCSs) and supervisory control and data acquisition (SCADA) networks [23]. In the energy sector, for example, local energy production processes in power plants employ DCSs, while SCADA networks are deployed for the power distribution to consumers spanning up to thousands of dispersed assets. For the explanations of process control networks, we use the generic term operational technology (OT), which describes hardware and software for controlling physical processes.

2.1 OT Network Hierarchy

The field of process control involves particular constraints and trends leading to major security issues that have already been identified for a decade from now [12], but which still unaltered exist due to the long-term design and operation of most networks. Regardless of the specific type, OT networks are typically divided into multiple sub-networks, resulting in an organisation in different layers. A simplified architecture of OT networks is presented in Fig. 1. On the lowest layer, i. e. the *fieldbus*, sensors and actuators measure and adjust physical parameters, e. g., pressure, temperature, or speed values, that are used by programmable logic controllers (PLCs) to fulfill a control-loop-based task. Data from several PLCs are collected in the *control network* for evaluation by servers that aggregate all activities of sub-processes and prepare these data for visualization. By connecting the (dual homed) servers with a second *supervisory network*, the data are made available there. Operators can interact with the processes via human-machine interfaces (HMI) by monitoring the data and manually adjusting sub-processes.

Nowadays, OT networks are connected to public networks, such as the Internet, to provide remote access to certain services provided by dedicated servers, e.g., data historians or maintenance servers, which are placed in a demilitarised zone *(DMZ network)*. Firewalls are used as major prevalent security measure to restrict access to these servers or to the OT network. To extend this basic protection, we propose monitoring *within* the subnets, which takes effect, for instance, when the firewall has been penetrated or an attack is launched within the network. The monitoring is based on a simple whitelisting of network communication. A set of rules defines the allowed communication behaviour, more precisely, which components are allowed to communicate with each other and how (with which protocols and message types).

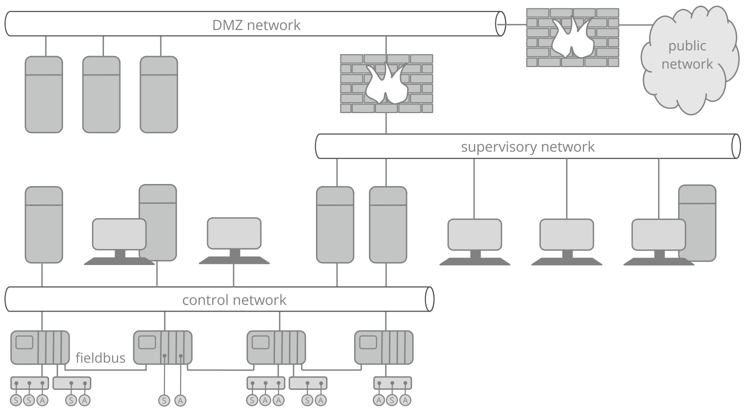


Fig. 1. Example of a typical OT network topology

2.2 Aspects of the Whitelist Use

Whitelisting has several advantages and disadvantages compared to alternative anomaly detection methods, which are discussed below regarding four aspects.

Completeness. Even in the case of a comprehensive learning, it cannot be guaranteed that a *complete* whitelist representing all legal network communication can be generated. An incomplete whitelist leads to the generation of false positives (FPs) that quickly cause an unacceptable administrative effort due to the necessary assessment of the alarms' relevance and the consideration of whitelist extensions. We conduct a detailed investigation of this in Sect. 5.

Monitoring Coverage. Whereas the completeness is associated with FPs, the monitoring coverage is linked to the false negatives (FNs) problem. Thus, attacks can occur in traffic parts not captured by the monitoring. Machine learning approaches in particular are usually limited to monitoring a specific traffic type, e.g., by targeted analysing process data or flow meta-information. In contrast, we follow a complete monitoring of the network traffic, independent of the protocol stack, up to the application-oriented protocol level.

Interpretability. Since the generation of FPs cannot be precluded, the application of systems that automatically initiate countermeasures, such as intrusion prevention systems, is usually not considered acceptable in OT networks. It is therefore more important that generated alerts are in a format that can easily be interpreted by a human being, so that a direct identification of the attack causes is possible. Compared to ML, whitelists offer great advantages here because the rule-based expression of detection references is inherently comprehensible.

Efficiency. Attack steps have to be logged at the shortest possible time interval after execution, so that countermeasures can immediately be initiated. It is likely that both the memory and the computing capacity required for the analysis are influenced by the size of the whitelist. Since the number of rules required for the desired monitoring cannot be estimated in advance, it is questionable whether a real-time monitoring can be achieved. We examine both the attack detection capability and the practical application of whitelisting in detail in Sect. 6.

3 Methodology

The proposed approach for whitelist generation from raw network traffic and the use for analysing the communication dynamics is shown in Fig. 2. Network traffic is first preprocessed to collect all information for specifying regular communications in the form of so-called *communication graphs* [19] (Sect. 3.1). A communication graph is first mapped to a *general whitelist* (Sect. 3.2) from which a *specific whitelist* (Sect. 3.3) is generated subsequently. The latter is adapted to a specific tool, here the IDS Snort [21], which can immediately be used for the monitoring. The whitelist application to analyse the communication dynamics and the experiments to investigate practical aspects are described in Sect. 5 and Sect. 6.

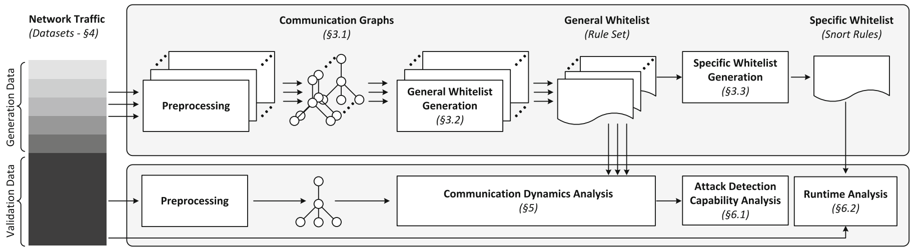


Fig. 2. Methodology overview

3.1 Communication Graphs

A communication graph is a directed graph used to represent network devices and their *logical* connections, also referred to as *communication relations*. Formally, a directed graph $D = (V, E)$ consists of a set of nodes (vertices)

$V = \{v_1, ..., v_n\}$ representing the network devices and a set of edges $E = \{e_1, ..., e_m\}$ for the communication relations. The addresses determined for a node v are represented by $A(v) = \{a \mid a \in \mathtt{A_{IP}} \cup \mathtt{A_{MAC}}\}$ with $\mathtt{A_{IP}}$ and $\mathtt{A_{MAC}}$ representing the domain of IP and MAC addresses. The set of all addresses of a communication graph D is referred to as $A(D)$:

$$A(D) = \bigcup_{i=1}^{n} A(v_i), v_i \in V(D)$$

An edge $e = (v_{src}, v_{dst}, M)$ contains the source and target vertices v_{src} and v_{dst} as well as a set of messages $M = \{m_1, ..., m_o\}$. A message $m = (a_{src}, a_{dst}, t, p, u)$ is characterised by the transmission protocol t, the payload protocol p, and the message type u. Since multiple addresses can be assigned to a vertex, the source and destination addresses used for message transmission are available in a_{src} and a_{dst}. We refer to all messages of a graph D as set $M(D)$:

$$M(D) = \bigcup_{i=1}^{m} M(e_i), e_i \in E(D)$$

A communication graph can directly be used as a reference to detect undesired communication. Next, we explain this procedure.

3.2 Generation of the General Whitelist

The message set $M(D)$ contains all information for describing legitimate network communications. Hence, it forms the initial rule set: $R = M(D)$. From this, more meaningful rules can be derived that focus on single aspects of communication. So, it is possible to distinguish between different types of whitelist violations.

Device-oriented Rules. These rules ensure that communication relations are exclusively established between already known devices. This includes both (1) detecting unknown devices and (2) new communication between known devices. The two objectives can be achieved by evaluating address information alone. Therefore, a device-oriented rule $r = (A_{src}, A_{dst})$ consists of only a 2-tuple for specifying legal communication partners. For the detection of unknown devices, a single rule $r_U = (A(D), A(D))$ is sufficient. To monitor the device-based communication of already known devices for each address $a \in A(D)$ the associated address set $A_{dst}(a)$ has to be determined. It is used to restrict regular target components that can be addressed by means of a:

$$A_{dst}(a) = \bigcup_{m \in M(D) | a_{src}(m) = a} a_{dst}(m)$$

In addition for each address $a \in A(D)$, a rule $r = (\{a\}, A_{dst}(a))$ is generated, resulting in the rule set R_K. We denote the rules for detecting new senders out of the set of already known devices as $R_{K_{src}} = \{r | r \in R_K \wedge A_{dst}(a) = \emptyset\}$. The set $R_{K_{dst}} = R_K \backslash R_{K_{src}}$ is used to identify devices that have already acted as senders but addressed different devices during monitoring. We refer to the complete set of device-oriented rules as $R_D = r_U \cup R_K$.

Communication-oriented Rules. After ensuring the legitimacy of device-to-device communication using the R_D rule set, specific communication type are verified. To generate these rules the tuple set $A_C(D)$ is first determined which contains all source and destination address combinations specified by graph D:

$$A_C(D) = \{a_C = (a_{src}, a_{dst}) | \bigcup_{m \in M(D)} (a_{src}(m), a_{dst}(m))\}$$

Each address tuple is used to determine the associated message set

$$M(a_C) = \{m \in M(D) | a_{src}(m) = a_{src}(a_C) \wedge \\ a_{dst}(m) = a_{dst}(a_C)\}$$

whose elements are subsequently combined in three different ways. First, communication protocols used for data transmission are determined, whereby transport-oriented protocols $T(a_C)$ and application-oriented protocols are distinguished, referred to as $P(a_C)$. In addition, the set of used message types $U(a_C)$ is determined for each application-oriented protocol. These three combination sets are formally described as follows:

$$T(a_C) = \bigcup_{m \in M(a_C)} t(m)$$

$$P(a_C) = \bigcup_{m \in M(a_C)} (t(m), p(m))$$

$$U(a_C) = \{((t,p), U) | (t,p) \in P(a_C), \\ U = \bigcup_{m \in M(a_C): t(m)=t \wedge p(m)=p} u(m)\}$$

A separate rule per address tuple is generated from each set:

$$R_T(a_C) = \{(a_{src}(a_C), a_{dst}(a_C), T(a_C))\}$$
$$R_P(a_C) = \{(a_{src}(a_C), a_{dst}(a_C), P(a_C))\}$$
$$R_U(a_C) = \{(a_{src}(a_C), a_{dst}(a_C), U(a_C))\}$$

We name the sets of all rules of these three rule types as R_T, R_P, and R_U, whose union gives the set of communication-oriented rules $R_C = R_T \cup R_P \cup R_U$.

3.3 Generation of Specific Whitelists

The rules describe aspects that are also analysed by existing security methods, such as firewalls or IDSs. Thus, it is straightforward to generate rules from the general whitelist for such tools. We realized that for Snort, a widely used IDS, by mapping the address information of the general rules to device-oriented rules and general communication-oriented rules to Snort header elements and options. We give a detailed demonstration of the generation principle of specific whitelists for Snort in respect of the syntax of Snort's description language in Appendix A.

4 Datasets

For the evaluation of our approach, we use eight datasets from different types of infrastructures that operate both OT networks and standard IT technology. Table 1 summarises the characteristics of the datasets. All are captured via dedicated access points (typically switches). The properties specified in Table 1 refer to the monitoring domain of the sensor used for capturing. As our evaluation is carried out from several aspects, the last column of the table contains information on whether a dataset is used for communication characterization *(com)*, attack detection capability evaluation *(det)*, or runtime analysis *(rta)*.

Table 1. Dataset characteristics

Dataset	Infrastructure type	Network level	Duration (hh:mm:ss)	# Packets (millions)	Packet rate (k/second)	# Devices	Analysis aspect		
							com	det	rta
power1.1	power plant	supervisory	02:39:34	90.53	9.46	114	●	○	●
power2.1		supervisory	02:15:36	66.08	8.12	71	●	○	○
power2.2		control	01:25:40	6.10	1.19	66	●	○	○
power2.3		DMZ	17:36:10	83.89	1.32	682	●	○	○
train1.1	train	control	01:35:44	17.00	2.96	76	●	○	○
train1.2		control	02:41:10	9.96	1.03	155	●	○	○
cicids.17	office	–	08:05:36	11.68	0.40	9,727	●	○	○
swat.a3	water treatment	control	24:12:58	1,248.96	14.00	61	●	○	○
swat.a6		control	03:40:00	321.03	24.00	98	○	●	●

Power Plants. We use several traffic traces that were captured at two coal-fired power plants. They involve two different control systems from leading vendors used worldwide. In our experience, networks of plants using the same control technology have strong similarities in architecture, protocols, and hardware. The dataset labeled as *power1.1* was captured within the supervisory level of a plant consisting of in total four generation units with a combined capacity of 2200 MW. The other datasets *power2.1* to *power2.3* originate from different network layers of a second plant with two generation units of the 800 MW class each.

Local Passenger Train. Two datasets originate from a control network of a train used for urban passenger transport. The train communication network is used for a train series rolled out in Germany at the end of 2018. The concrete train is composed of six coaches including two railcars. The operation is implemented by a network (conform to [5]) spanning all coaches. It is divided into two local networks. The first subnet contains devices of subsystems that are essential to ensure regular train operation. These include, for example, traction/brake and door control, power converters, and safety systems. The second subnet includes devices of various systems whose fault-free operation does not have a decisive influence on the safe operation of the train. They serve organisational and coordination purposes or contribute to passenger comfort. These include the camera surveillance system, passenger counting, intercoms and passenger information devices. Both subnets reflect the control network layer.

Water Treatment. This dataset [8][3] represents a small-scale version of a modern water treatment plant. The network traffic considered in this study was provided in two different datasets. The first one originates from 2016 and covers 136 h of network traffic from continuously running SWaT without performing any attacks. We use the first about 24 h of this data and refer to them as *swat.a3*. The dataset *swat.a6* was provided in 2019 and contains a series of malware infection attacks on the engineering workstation. Due to the network architecture of the testbed, the two datasets are assigned to the control level.

Standard IT Network. In addition to the OT traffic, we use a portion of the popular public dataset CICIDS2017 [22] as an exemplary representative of standard IT traffic for a direct comparison of communication characteristics. The dataset labeled *cicids.17* corresponds to the first day (Monday) of the capture when no attack activity has been observed.

Notes on Private Datasets. The first two datasets contain sensitive information that we are only allowed to describe in abstract form due to non-disclosure agreements. The datasets were captured using the tool `tcpdump`, running on a separate sensor device attached to switches with activated port mirroring.

5 Communication Dynamics Analysis

Besides the whitelist generation, the process depicted in Fig. 2 aims to analyse the communication dynamics of OT networks. Here, we focus on investigating the variability or homogeneity of the traffic and on determining whether (and to what extent) a complete whitelist can be generated from it.

5.1 Multi-step Whitelist Generation

For the communication dynamics analysis, each dataset is first divided into two parts, the *generation data* and the *validation data.* The whitelist is generated in an n-step procedure in which the generation data is further split into n disjoint and consecutive sub-captures. In each step i $(1 \leq i \leq n)$, an increasing amount (i sub-captures) of the generation data is used for the whitelist generation. After each step, the validation data is analysed with the derived whitelist and the number of packets that do not match the whitelist is determined. This proportion of packets from the total amount of validation data is called *mismatching packet rate (MPR).* The MPR in the i-th step is denoted as m_i.

5.2 Measures for MPR Evolution Analysis

The MPR determined in the last sub-step m_n is an indicator of the completeness of the whitelist generated from the generation data used to monitor the validation data. To characterize the communication dynamics within the generation data,

[3] https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/.

multiple measures are determined. The basis is the MPR reduction achieved between two sub-steps. Due to the monotonicity of the MPR evolution function, the MPR decrease after the i-th step is determined as difference of the current MPR and the MPR of the next step: $d_i = m_i - m_{i+1}$, with $1 \leq i < n$.

Mean MPR Decrease. As base for the following measures this is defined by:

$$\bar{d} = \frac{1}{n-1} \sum_{i=1}^{n-1} d_i$$

Dispersion Measures. In addition to the mean MPR decrease, the communication dynamics can also be characterized by the distribution around the mean value. A higher distribution indicates significant whitelist extensions within relatively few generation steps. Thus, communication can be considered more static in the case of a comparatively high dispersion than in the case of a relatively low one. The analysis of the dispersion of individual MPR reduction rates around the mean is based on relative measures. First, the relative standard deviation is determined, which is also referred to as the variation coefficient:

$$v = \frac{\sqrt{\frac{1}{n-1} \sum_{i=1}^{n-1} (d_i - \bar{d})^2}}{\bar{d}}$$

As a second measure of dispersion, the normalized Gini coefficient is determined. For this purpose, the MPR decrease values are first sorted in increasing order; we denote the values thus obtained as $\hat{d}_1, ..., \hat{d}_{n-1}$, with $\hat{d}_i \leq \hat{d}_{i+1}$. The normalized Gini coefficient is determined from these as follows:

$$g = \left(\frac{2 \cdot \sum_{i=1}^{n-1} i \cdot \hat{d}_i}{n - 1 \cdot \sum_{i=1}^{n-1} \hat{d}_i} - \frac{n}{n-1} \right) \cdot \frac{n-1}{n-2}$$

For the normalized Gini coefficient, it holds $0 \leq g \leq 1$, with the value rising as the inequality dispersion increases. Finally, we introduce a measure to analyse how often the whitelist changes *significantly* during generation. Hence, we define the jump rate j as the fraction of decreasing MPR values that exceed a certain threshold. We use the product of the mean MPR decrease and the variation coefficient as the relative threshold, so that the jump rate is defined by:

$$j = \frac{|\{d | d \in \{d_1, ..., d_{n-1}\} \wedge d > (\bar{d} \cdot v)\}|}{n-1}$$

5.3 Experimental Setup and Results

The whitelist generation process (Sect. 5.1) is applied as follows. The split of a dataset into generation and validation data was performed in a 1:1 ratio and ten sub-steps ($n = 10$) were used to create the whitelist from the generation data. All splits are time-based, so each resulting sub-capture has the same duration.

Table 2. Whitelist information, mismatching packets, and clustered MPR evolution measures for the investigated datasets

Dataset	**Network level**	**# Triggered rules / # Total rules**						**# Mism. packets**	**MPR evolution**				
		device-oriented			*comm.-oriented*								
		r_U	$R_{K_{src}}$	$R_{K_{dst}}$	R_T	R_P	R_U		m_{10}	$\bar{d}$	v	g	j
power1.1	superv.	$\frac{1}{1}$	$\frac{0}{10}$	$\frac{9}{54}$	$\frac{0}{151}$	$\frac{3}{151}$	-	336	0.000721	0.000532	1.697286	0.827277	$0.\bar{2}$
							2/151	47,604	0.102202	0.000534	1.686094	0.819754	$0.\bar{2}$
power2.1	superv.	$\frac{1}{1}$	$\frac{0}{9}$	$\frac{13}{83}$	$\frac{2}{226}$	$\frac{8}{226}$	-	366,176	1.166194	0.000731	1.388637	0.772760	$0.\bar{3}$
							1/226	367,188	1.169417	0.000731	1.388637	0.772760	$0.\bar{3}$
power2.2	control	$\frac{0}{1}$	$\frac{0}{5}$	$\frac{1}{68}$	$\frac{0}{222}$	$\frac{0}{222}$	-	27	0.000917	0.005911	2.747624	0.991066	$0.\bar{1}$
							2/222	31	0.001052	0.005911	2.747624	0.991066	$0.\bar{1}$
power2.3	DMZ	$\frac{1}{1}$	$\frac{19}{109}$	$\frac{96}{514}$	$\frac{18}{3028}$	$\frac{31}{3028}$	-	9,146,857	19.463888	0.569613	2.312673	0.946033	$0.\bar{1}$
							15/3028	9,187,419	19.550202	0.570594	2.309209	0.945694	$0.\bar{1}$
train1.1	control	$\frac{0}{1}$	$\frac{0}{34}$	$\frac{0}{67}$	$\frac{0}{207}$	$\frac{0}{207}$	-	0	0.0	0.000631	2.649324	0.985537	$0.\bar{1}$
							0/207	0	0.0	0.000631	2.649324	0.985537	$0.\bar{1}$
train1.2	control	$\frac{1}{1}$	$\frac{1}{50}$	$\frac{12}{123}$	$\frac{1}{270}$	$\frac{1}{270}$	-	6,252	0.126118	0.397197	2.821426	0.998874	$0.\bar{1}$
							0/270	6,252	0.126118	0.397197	2.821426	0.998874	$0.\bar{1}$
swat.a3	control	$\frac{1}{1}$	$\frac{0}{21}$	$\frac{8}{53}$	$\frac{1}{272}$	$\frac{0}{272}$	-	57	0.000009	0.000177	2.797246	0.996615	$0.\bar{1}$
							6/272	64	0.000011	0.008177	2.827717	0.999923	$0.\bar{1}$
cicids.17	-	$\frac{1}{1}$	$\frac{1}{40}$	$\frac{2{,}796}{7{,}065}$	$\frac{88}{27{,}145}$	$\frac{1{,}326}{27{,}145}$	-	1,592,881	54.123183	2.941505	0.847902	0.477112	$0.\bar{4}$
							16/27,145	1,593,120	54.131304	2.940656	0.848042	0.477112	$0.\bar{4}$

The measures introduced in Sect. 5.2 for the individual datasets are presented in Table 2[4] Since the traffic of different networks exhibits a different protocol mix and not every protocol is decoded to distinguish message types, two values per measure were determined for each dataset. The lower line shows the measures that takes the rules for the identification of different message types into account. In comparison, the upper row states the measures from the exclusive application of the remaining rule types. For allowing a quick visual comparison among the datasets, the results are clustered (column-wise) into four coloured groups (corresponding to the number of different network layers) using k-means. Table 2 also states for each rule type (cf. Sect. 3.2) the ratio of rules triggered by mismatching packets to the total number of generated rules. Thus, the sum of the numerators corresponds to the number of rules that would have to be extended to complete the whitelist with respect to a mismatching packet free analysis of the validation data. The evolution of the MPR over the whitelist generation steps and the influence of the different rule classes on the MPR is shown in the bar charts in Fig. 3.

5.4 Characterization Takeaways

Finding 1: Communication Whitelisting is not Suitable for Monitoring Standard IT Networks. With an MPR of over 54%, by far the most incomplete whitelist was generated for the *cicdis.17* dataset. Due to about 4,000 different rules to be extended, it must be concluded that a whitelist cannot be developed with proper effort to a complete one. The values determined for the dispersion measures indicate a uniform development of the MPR reduction and thus a

[4] A repository for interactive analysis of the public datasets is provided here: https://gitlab.com/paulandre/ot-whitelisting.

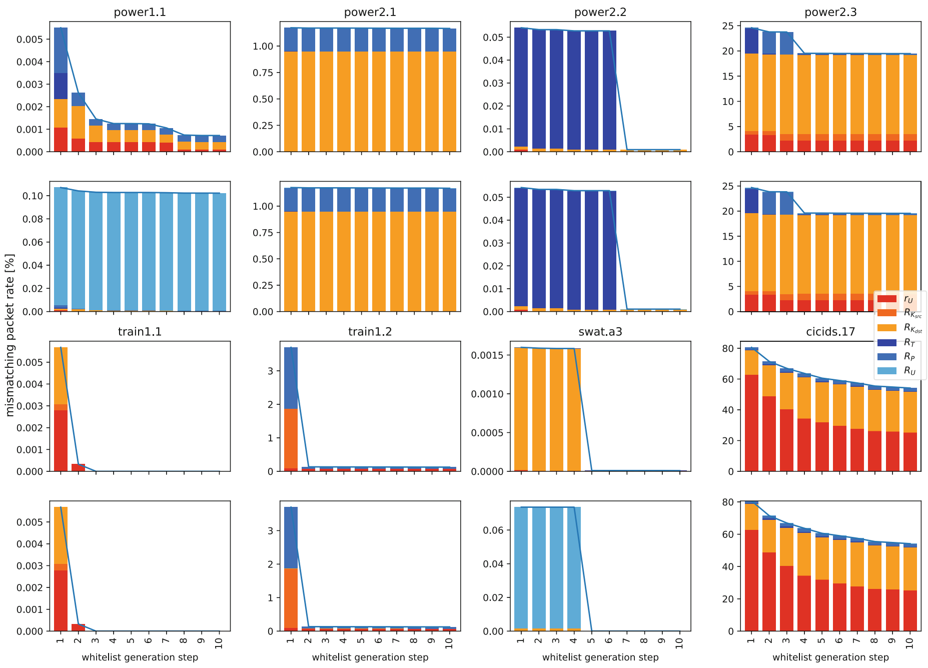


Fig. 3. Evolution of MPRs taking (rows two and four) and not taking (rows one and three) message-type-specific rules into account

high communication dynamics in the generation data. Consequently, we consider communication whitelisting to be unsuitable for standard IT traffic represented by the *cicdis.17* dataset and do not discuss further details for this dataset.

Finding 2: Different OT Network Layers Exhibit Different (measurable) Communication Dynamics. A clustering of dispersion measures shows that communication dynamics differ significantly among OT network layers. Already on the basis of the jump rate measure, the supervisory networks ($j = 0.\overline{2}$, resp. $j = 0.\overline{3}$) can be distinguished from other layers ($j = 0.\overline{1}$). However, as the range of the jump rate is too small, this measure is not suitable for an accurate classification of network domains. In contrast, both the coefficient of variation and the Gini coefficient can be used to clearly refer traffic to OT network layers. Furthermore, the communication dynamics measures indicate that the lower the network layer, the smaller the differences in communication dynamics among different networks of the same layer. While the large difference between the two datasets representing the supervisory layer can even be discerned in the jump rate, the control-layer networks show a comparatively small range in terms of the coefficient of variation (2.65 to 2.83) and the Gini coefficient (0.986 to 0.999).

Finding 3: Strong Correlation between Communication Dynamics and Whitelist Completion Effort. Looking at the communication dynamics in isolation using the measures to describe MPR evolution does not yet allow any conclusions to be drawn about the effort required to extend the whitelist towards a complete one, since in the extreme cases to detect n mismatching packets exactly one, but also n rules may be responsible. By including information on triggering rules, a relatively strong negative correlation can be determined between the proportion of triggering rules from the total amount of rules (r_t for short) and the MPR dispersion measures. Depending on the consideration or exclusion of the message-type-specific rules, a correlation coefficient of -0.79 and -0.83 was determined between r_t and v, respectively. The correlation coefficient between r_t and g is -0.81 (taking message-type-specific rules into account) and -0.84 (when excluding these rules), respectively.

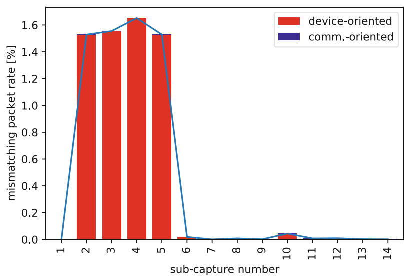
Finding 4: Detection of Whitelist Violations is Dominated by Device-oriented Rules. New device-level communication relations are the most common cause of logging whitelist mismatching packets. Only for dataset *power1.1*, communication-specific rules are dominant, in case message-type-specific rules are considered. For the remaining five datasets, the percentage of messages logged by device-oriented rules ranges from 61% *(train1.2)* to 98% *(power2.3)*. The most common cause of new device-level communication relations involves, between known components, a component already acting as a sender addressing one or more additional devices.

6 Whitelist Application

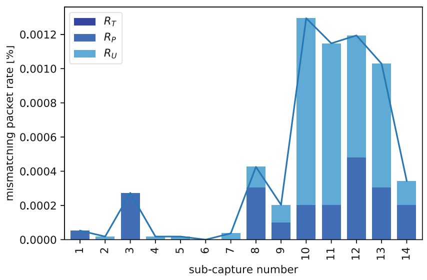
We discuss the application by detection capability and evaluation performance.

6.1 Attack Detection Capability

The attack detection capability of the whitelists is exemplarily evaluated using the *swat.a6* dataset containing four malicious events. Information on the allocation of these events to the corresponding sub-captures are taken from the documentation of the dataset. A whitelist was generated from the first sub-capture (referred to as *c0*), which does not contain any malicious traffic. Figure 4(a) shows the MPRs derived from the remaining sub-captures *c1-c14*.



(a) Total MPR



(b) Communication-based MPR

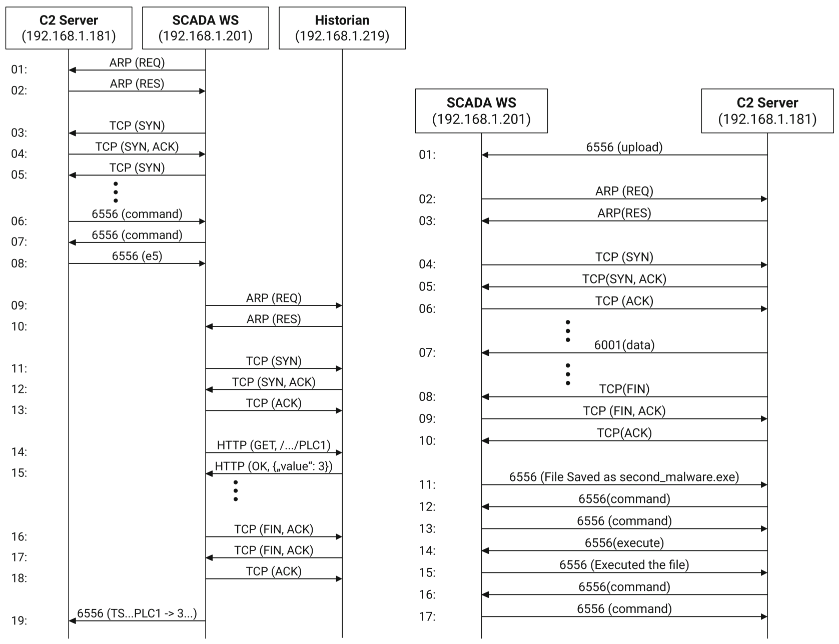
Fig. 4. Mismatching packet rates of the SWaT A6 sub-captures

Infiltrate SCADA WS via USB Thumb Drive with First Malware. A first malware infection is assigned to *c1*. As the pure infection does not immediately result in changed communication relations, a relatively low MPR (0.000139%) was determined for this capture.

Exfiltrate Historian Data. In a total of four cycles, which are assigned to *c2* to *c5*, a data exfiltration attack was performed. These result in high MPRs (>1.5%), whereby logged packets mainly belong to the communication of three devices. Figure 5(a) shows a message sequence chart (MSC) representing their communication. The malware apparently enables a remote control of the infected host. The SCADA workstation first establishes a TCP connection on port 6556 (messages *3* to *5*, *m3* to *m5* for short) to the device designated as command-and-control (C2) server. After exchanging initial alive messages, the C2 server transmits a command (e5, see *m8*) to the workstation, whereupon the workstation requests current process data from the historian. For this purpose, a TCP connection on port 8080 is established, transferring HTTP packets to request files from the historian *(m14)*. Subsequently, the historian responds *(m15)*, transferring the file content in JSON format. This polls the current measurement and status values of all devices involved in the six sub-processes step by step. Afterwards, the data is forwarded with a single TCP packet *(m19)*. The data polling from the historian and the transmission of the collected values to the C2 server is repeated at regular intervals of about one second. The logging of the associated packets was caused entirely by device-oriented rules. More precisely, the communication between the victim and the C2 server (a so far unknown device) was detected by r_U. In contrast, the communication between the victim and the historian was identified by two rules from set $R_{K_{dst}}$.

Infiltrate SCADA WS with Second Malware, via Downloading from C2 Server. After a rest period of 60 min, represented by *c6* to *c9*, a secondary infection of the workstation occurs by reloading software from the C2 server. This event is assigned to *c10*, where a slightly increased MPR (∼0.044%) can be observed, that is mainly caused by the communication between the workstation and the C2 server. Packets violating the whitelist are summarised by the MSC shown in Fig. 5(b). This reveals that the malware operates through two different channels. According to the execution of the data exfiltration attack, the control commands are transmitted over a TCP connection on port 6556. In addition, a connection established on port 6001 serves as a data channel. After the *upload* command has been received *(m1)*, the workstation initiates the establishment of the data connection. Afterwards, the C2 server transfers the data to the victim by sending several packets. Once the data transfer is finished, the TCP connection is closed (*m8* to *m10*). From the commands that are subsequently exchanged over the second channel, it can be concluded that the transmitted data is the malware executable file that is executed after it is stored.

Disrupt Sensor and Actuator. The malware execution tends to sensor and actuator disruption, performed in five cycles associated to *c10* to *c13*. While the visibly increased MPR to *c10* is caused by the malware transmission (Sect. 6.1),



(a) Data exfiltration (b) Malware upload

Fig. 5. Sequence charts representing analysed attacks

further events cannot be identified from the overall MPR. However, when looking exclusively at the results of the communication-oriented rules (see Fig. 4(b)), a significantly increased MPR for the sub-captures is evident. Here, the MPR is dominated by rules of the set R_U used to detect new message types. For *c10*, a total of 236 associated packets can primarily be assigned to a communication between the workstation and the primary PLC used to control the first sub-process (P1) of the SWaT's six-stage process. The logged communication between these components consists of 220 messages transmitted via EtherNet/IP combined with the Common Industrial Protocol (CIP). There are a combined total of ten different message types that violate the whitelist. Figure 6 shows the communication. First, a session is established by means of EtherNet/IP *(m1, m2)*. The application-based communication between the victim and PLC is realised using CIP, also operating in a connection-oriented manner. A two-way handshake is used to establish and terminate the connection (*m5+m6* and *m9+m10*, respectively). To ensure that no path to the objects to be accessed exists yet, prior to this *(m3, m4)* an attempt is made to terminate the connection. Object access by the workstation is executed by *m7* and confirmed by the PLC through *m8*.

6.2 Application of a Specific Whitelist

Experimental Setup. We perform a communication monitoring with a runtime analysis using Snort as an example. Here, we focus on *power1.1* and *swat.a6* as the private and public dataset with the highest packet rates. Snort rules were generated from the general whitelists created to measure communication dynamics (in the case of *power1.1*) or to analyse attack detection capabilities *(swat.a6)*.

Table 3 summarises the mean packet processing rates for ten runs of Snort using certain rule types as well as the complete rule set. In case of *power1.1* two sets of validation data were applied. First, the complete whitelist generation traffic *(val1)* was used, which naturally leads to no alarms. For comparing processing rates with and without alarms the priorly defined validation part *val2* used to measure the communication dynamics was also monitored. Regarding *swat.a6*, the validation data corresponds to the traffic analysed in Sect. 6.1.

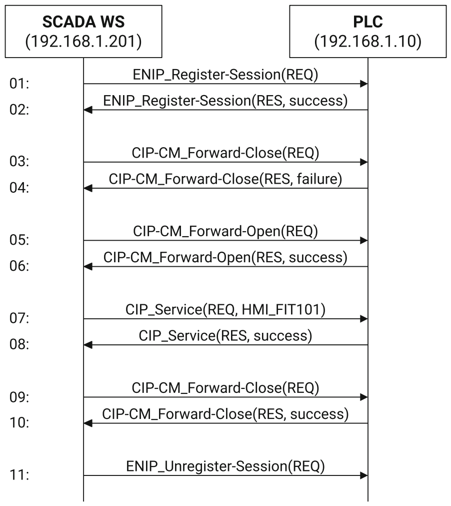

SCADA WS (192.168.1.201)
PLC (192.168.1.10)
01: ENIP_Register-Session(REQ)
02: ENIP_Register-Session(RES, success)
03: CIP-CM_Forward-Close(REQ)
04: CIP-CM_Forward-Close(RES, failure)
05: CIP-CM_Forward-Open(REQ)
06: CIP-CM_Forward-Open(RES, success)
07: CIP_Service(REQ, HMI_FIT101)
08: CIP_Service(RES, success)
09: CIP-CM_Forward-Close(REQ)
10: CIP-CM_Forward-Close(RES, success)
11: ENIP_Unregister-Session(REQ)

Fig. 6. Sequence chart of process disruption

Table 3. Packet processing rates using Snort

		power1.1		*swat.a6*
		val1	val2	
device-orient.	#rules	43		57
	#alerts	0	87	1,391,094
	10^3 pkts/s	762	773	615
R_T	#rules	144		219
	#alerts	0	0	0
	10^3 pkts/s	563	574	432
R_P	#rules	161		140
	#alerts	0	0	192
	10^3 pkts/s	364	375	294
R_U	#rules	976		22
	#alerts	0	46,126	0
	10^3 pkts/s	2,429	2,416	3,644
all	#rules	1,208		435
	#alerts	0	46,213	1,391,286
	10^3 pkts/s	225	225	113

Performance Results. All rule types allow processing rates in the three-digit thousands range. The rate decreases with an increasing amount of packet header information that has to be analysed. A significantly high processing rate was achieved when only message-type-specific rules were used. This can be explained by the fact that only for a subset of the communication protocols a distinction between different message types is made. Depending on the protocol mix of the analysed data, there is only a very small proportion of end-to-end communication relations (specified by the header of the Snort rules) whose packets need to be analysed at all, so that the analysis of the majority of packets can be terminated at a very early stage. The rule set R_U is significantly smaller in the case of the *swat.a6* dataset, which recognisably results in a higher processing rate. Since there are no relevant differences in the analysis performance of the two *power1.1* validation datasets, we conclude that only the number of rules used is decisive and that the logging effort is of no importance. As all packet processing rates exceed the communication packet rates determined for the datasets (cf. Table 1) by a factor of 4.5 to 23.5, we conclude that our whitelists can be used for a real-time monitoring of OT traffic.

7 Related Work

We aim to contribute to OT security by both characterizing OT traffic and providing a monitoring technique according to these characteristics. Hence, we discuss the state of the art in both areas.

7.1 Characterization of Process Control Traffic

We distinguish two groups for characterizing process control traffic and refer to the terms SCADA and DCS as differentiated in [23].

Aggregated Traffic Characterization. This kind focuses on the quantification of traffic meta information, which is motivated by the need for realistic traffic simulation in research. For SCADA networks, the approaches outline differences to standard IT networks by measuring the periodicity in terms of the frame rate and the number of active connections [2] and investigate the application of standard IT traffic models, which are figured out as not transferable to SCADA traffic [3]. Other research measured the TCP-based DNP3 communication in terms of polling intervals, inter-arrival times, idle and round-trip times per device, (temporal and byte) duration of TCP flows, retransmission rates and timeouts [7]. Other works studied the traffic of the SCADA protocol IEC 60870-5-104 by categorizing the traffic into strongly cyclic, weakly cyclic, stable, bursty, and phase transitional by quantifying distribution changes of event inter-arrival times over time [15,16]. The authors of [13] recently proposed a five-step method for profiling traffic by quantifying communication intensity, recognizing work-cycles by repeated communication patterns, and identifying the work-cycles' states with their subsequent profiling. Regarding DCS, the presence of a rich protocol mix has recently been identified in [18], in contrast to what is stated for SCADA networks. The authors show the feasibility to distinguish proprietary protocols by clustering traffic based on the inter-arrival times and header data of frames.

Structural Traffic Profiling. Related approaches try to model periodic communication patterns. While this was initially examined for supporting single protocols [9,10], other works proposed a protocol-independent modelling of concrete periodic traffic patterns [4,11] exploring message repetition and timing information. They evaluated their approaches for a DCS (Siemens S7 and MMS, respectively) as well as for a SCADA protocol (both Modbus). An analysis of the IEC 60870-5-104 communication of the power distribution backbone network is available in [17]. In addition to the analysis of physical network changes and the communication flow lengths of the SCADA infrastructure, particularly structural traffic analyses are presented in the form of clustering session variants and measurements of the amount and the semantics of message types. A recent publication [6] suggest a deep-packet inspection on OT protocols to generate models of communications between network device pairs, what they demonstrate for three protocols (Modbus/TCP, DNP3, and EtherNet/IP) for SCADA and DCS networks. They, however, admit that the used Discrete Time Markov Chains suffer from state and transition explosion.

Research Gap. The former type of characterizations abstract network activities to aggregated traffic observations. These allows only a topview on OT traffic quantities. The latter type of works analyzes communication frames, but it dives to deep into the perspective of the process (modelling of process variables) and thus loses track of network transactions and runs into state explosion. We target at the gap between the two trends providing (1) a bottom view on the traffic by analyzing frames instead of aggregated traffic properties, (2) by keeping the network perspective as prerequisite for a network operator's understanding and re-use of findings, and (3) by providing a profiling approach capable to incorporate a mix of protocols in the network, as it was figured out to be the case especially for DCS [18] and so-called brownfield systems [13].

7.2 Attack Detection for OT

For the sake of brevity we refer to the recent structured overview given in [25]. Our work falls into the smaller group of communication-based approaches, which has decisive advantages over process state-aware approaches. These include the applicability on networks independent from the existence of (potential vendor-specific) systems for the necessary preprocessing to provide well-structured sensor data. Many detection approaches have been identified as unnecessarily complex [24] which relates to serious problems of interpretability and reliability in real-world scenarios. In contrast, the presented approach provides both, transparency into the modeled detection knowledge in the form of precise rules as well as the possibility to influence the detection by the adaptions of the rules.

8 Conclusion

In this paper, we presented a whitelisting approach for characterizing and monitoring communication in OT networks. We used it for measuring eight datasets to express the homogeneity of OT traffic and differences among OT network layers. We examined that the whitelists meet essential criteria regarding completeness (measured by a *mismatching packet rate*), interpretability (inherent to rule-based approaches), detection capability (identified for a prominent OT dataset), and efficiency (measuring the packet processing rates when using the whitelists with a well-known IDS). Although the generation cannot guarantee complete whitelists even in the OT domain, which can lead to false alarms, the evaluation still proves the whitelists are very effective for process control traffic. Hence, they can serve as an interpretable baseline in the OT domain in order to justify the composition of more complex methods, which is often neglected when proposing new detection schemes [1]. Consequently, up next we will use the whitelist approach to define baselines for the OT domain using public data (e. g., [14,20]) and to transpose current rough time-based packet labelings of public datasets into a precise pattern-based one.

A Specific Whitelist's Generation for Snort

As a prerequisite to follow the description, an exemplary Snort rule is shown in Fig. 7.

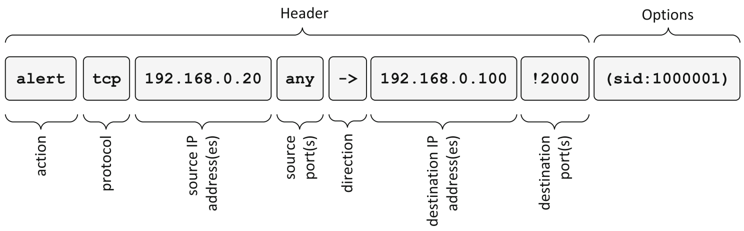


Fig. 7. Example of a Snort rule

Device-orientated Rules. Device-oriented Snort rules are created from the address information of the general rules of this class. Since only IP-based network traffic can be specified by Snort rules, all MAC addresses have to be removed from the applied address sets. A set of addresses A filtered by IP addresses is called $A' = \{a | a \in A \cap \mathtt{A_{IP}}\}$. The mapping of the general rule r_U is done using the set $A(D)' = \{a_1, ..., a_i\}$ as follows:

$$\texttt{alert ip !}[a_1, ..., a_i]\texttt{ any -> any any}$$

As Snort operates in blacklist mode, the exclamation mark as negation operator is used which causes the rule to trigger an alert whenever a device with an unknown IP address acts as a sender. The keyword `any` specifies the complete value range of the respective header element (destination IP addresses, and source/destination ports). Accordingly, the mapping of a rule $r \in R_K$ with a source address $a \in A(D)'$ and the corresponding address set $A_{dst}(a)'$ is done accordingly to (1) in case $A_{dst}(a)' \neq \emptyset$, otherwise to (2):

$$(1)\ \texttt{alert ip } a \texttt{ any -> !}[a_1, ..., a_j]\texttt{ any}$$
$$(2)\ \texttt{alert ip } a \texttt{ any -> any any}$$

Communication-orientated Rules. To map general communication-oriented rules, Snort header elements and Snort options are used. Since only rules for the specification of IP-based communication can be mapped here as well, further notations of the sets used for the mapping are given first. We denote the set of address tuples $A_C(D)$ filtered by IP addresses as $A_C(D)'$, the set of messages determined for an address tuple $a_C{}'$ as $M(a_C{}')$, and the aggregated sets from these as $T(a_C{}')$, $P(a_C{}')$, and $U(a_C{}')$, respectively. For a definition of allowed transport-oriented protocols, the `protocol` field of the Snort header is used. Since the application of the negation operator is not provided for this element and also no list can be used, the mapping is done by several Snort rules if necessary. To this end, the set of transport protocols to be mapped $T_M = \{t | t \in T_S \setminus T(a_C{}')\}$ is determined, where $T_S = \{ip, icmp, tcp, udp\}$ corresponds to the set of protocols that can be used in the header. A Snort rule is created for each $t \in T_M$:

$$\texttt{alert}\ t\ a_{src}\ \texttt{any -> }a_{dst}\ \texttt{any}$$

The generation of Snort rules for the specification of legal application-oriented protocols is basically done by mapping the set $P(a_C{}')$ to the `protocol` header element as well as the fields `source_ports` and `dest_ports`. The first step here is to create a set that contains all port numbers in combination with the transport protocol. It specifies the set of legitimate application protocols:

$$P_S(P(a_C{}')) = \{(t, P) | t \in t(p_C \in P(a_C{}')) \cap \{tcp, udp\},$$
$$P = \bigcup_{p_C \in P(a_C{}') : t(p_C) = t} p(p_C)\}$$

However, a Snort rule can only be created from this set if there is a distinct client-server relationship between the communication partners regarding the packets to be specified by the rule. If the corresponding services are hosted by the destination component the mapping is as follows:

$$\texttt{alert}\ t\ a_{src}\ \texttt{any -> }a_{dst}![p_1, ..., p_n]$$

If the services are provided by the source device the values of the `source_ports` and `dest_ports` header fields are swapped. To determine the server component of a specific service the following strategies with descending priority are used:

1. Message types: For typical client-server-oriented services, the observed message types provides immediate information about the device role. For example, in the case of DNS, when a request message is detected, the sender is considered as a client and the target device as a server.
2. Connection establishment: In case of TCP-oriented services, an observed connection establishment can be used for the assignment, whereby for the first (or third) packet of the three-way handshake the sender is considered as the client and the destination as the server.
3. Heuristic: The heuristic role determination is based on the subgraph resulting from the communication graph filtered by the protocol used to transmit the messages used for service provision. If a service is used by multiple clients the server can be identified by the associated node in the graph with the highest node degree (indegree and outdegree).

Since the differntiation of message types relies on the protocol-specific decoding of packet payload data, there is no generalized way to map message types specified by general rules to a set of Snort rules. We present two exemplary ways.

Preprocessor Usage: Some existing preprocessors already allow explicit logging of packets with specific message types being transmitted. For Modbus packets, for example, the option `modbus_func` (Modbus function code) is available for this purpose. Since negation and the specification of multiple values are also not provided when using this option, a rule must be created for each non-permitted Modbus message type. For example, packets sent from a_{src} to a_{dst} to request coils status are logged by the following rule:

alert tcp a_{src} any -> a_{dst} 502 (modbus_func:read_coils)

Use of Payload Detection Options: Snort can be extended by further preprocessors for the detection of message types of arbitrary protocols. However, since the proposed method is primarily intended to support existing unmodified tools, the development of any extensions is omitted. Because the message type is often encoded in the first bytes of the payload, another way to detect different message types is to use options for payload investigation, such as `content` and `byte_test`, which selectively examine byte values for one given pattern. Another possibility is provided by the `pcre` option which can be used to specify a set of patterns of illegitimate message types by means of a regular expression, thus requiring only one rule to be created per PDU-specific communication relation. If, for example, all read requests from a_{src} to a_{dst} should be logged, the following rule can be used as an alternative to four different rules using the `modbus_func` option:

alert tcp a_{src} any -> a_{dst} 502
`(pcre:"/^.{7}(\x01|\x02|\x03|\x04)/s")`

References

1. Arp, D., et al.: Dos and don'ts of machine learning in computer security. In: USENIX Security Symposium. USENIX Association (2022)
2. Barbosa, R.R.R., Sadre, R., Pras, A.: A first look into SCADA network traffic. In: Network Operations and Management Symposium (NOMS). IEEE (2012)
3. Barbosa, R.R.R., Sadre, R., Pras, A.: Difficulties in modeling SCADA traffic: a comparative analysis. In: Taft, N., Ricciato, F. (eds.) PAM 2012. LNCS, vol. 7192, pp. 126–135. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28537-0_13
4. Barbosa, R.R.R., Sadre, R., Pras, A.: Exploiting traffic periodicity in industrial control networks. Int. J. Crit. Infrastruct. Prot. **13** (2016)
5. Commission, I.E.: IEC 61375–1:2012 Electronic railway equipment - Train communication network (TCN) - Part 1: General architecture (2012)
6. Faisal, M.A., Cardenas, A.A., Wool, A.: Profiling communications in industrial IP networks: model complexity and anomaly detection. In: Alcaraz, C. (ed.) Security and Privacy Trends in the Industrial Internet of Things. ASTSA, pp. 139–160. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12330-7_7

7. Formby, D., Walid, A.I., Beyah, R.A.: A case study in power substation network dynamics. In: International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS). ACM (2017)
8. Goh, J., Adepu, S., Junejo, K.N., Mathur, A.: A dataset to support research in the design of secure water treatment systems. In: Havarneanu, G., Setola, R., Nassopoulos, H., Wolthusen, S. (eds.) CRITIS 2016. LNCS, vol. 10242, pp. 88–99. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71368-7_8
9. Goldenberg, N., Wool, A.: Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. Int. J. Crit. Infrastruct. Prot. **6**(2), 63–75 (2013)
10. Kleinmann, A., Wool, A.: Accurate modeling of the siemens S7 SCADA protocol for intrusion detection and digital forensic. J. Digit. Forensics Secur. Law **9**(2), 37–50 (2014)
11. Kleinmann, A., Wool, A.: Automatic construction of statechart-based anomaly detection models for multi-threaded SCADA via spectral analysis. In: Workshop on Cyber-Physical Systems Security and Privacy (CPS-SPC). ACM (2016)
12. Krotofil, M., Gollmann, D.: Industrial control systems security: what is happening? In: International Conference on Industrial Informatics (INDIN). IEEE (2013)
13. Lavassani, M., Åkerberg, J., Björkman, M.: Modeling and profiling of aggregated industrial network traffic. Appl. Sci. **12**(2) (2022)
14. Lemay, A., Fernandez, J.M.: Providing SCADA network data sets for intrusion detection research. In: Workshop on Cyber Security Experimentation and Test (CSET). USENIX Association (2016)
15. Lin, C., Nadjm-Tehrani, S.: Understanding IEC-60870-5-104 traffic patterns in SCADA networks. In: Workshop on Cyber-Physical System Security (CPSS). ACM (2018)
16. Lin, C., Nadjm-Tehrani, S.: Timing patterns and correlations in spontaneous SCADA traffic for anomaly detection. In: International Symposium on Research in Attacks, Intrusions and Defenses (RAID). USENIX Association (2019)
17. Mai, K., Qin, X., Silva, N.O., Molina, J., Cárdenas, A.A.: Uncharted Networks: a first measurement study of the bulk power system. In: Internet Measurement Conference (IMC). ACM (2020)
18. Mehner, S., Schuster, F., Hohlfeld, O.: Lights on power plant control networks. In: Hohlfeld, O., Moura, G., Pelsser, C. (eds.) PAM 2022. LNCS, vol. 13210, pp. 470–484. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-98785-5_21
19. Paul, A., Schuster, F., König, H.: Network topology exploration for industrial networks. In: Maglaras, L.A., Janicke, H., Jones, K. (eds.) INISCOM 2016. LNICST, vol. 188, pp. 62–76. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52569-3_6
20. Rodofile, N.R., Schmidt, T., Sherry, S.T., Djamaludin, C., Radke, K., Foo, E.: Process control cyber-attacks and labelled datasets on s7comm critical infrastructure. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10343, pp. 452–459. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59870-3_30
21. Roesch, M.: Snort: lightweight intrusion detection for networks. In: Conference on Systems Administration (LISA). USENIX (1999)
22. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: International Conference on Information Systems Security & Privacy (ICISSP). SciTePress (2018)
23. Stouffer, K., Pillitteri, V., Lightman, S., Abrams, M., Hahn, A.: Guide to industrial control systems (ICS) security. NIST Spec. Publ. 800–82 (2015). Rev. 2

24. Wolsing, K., Thiemt, L., Sloun, C.v., Wagner, E., Wehrle, K., Henze, M.: Can industrial intrusion detection be SIMPLE?. In: Atluri, V., Di Pietro, R., Jensen, C.D., Meng, W. (eds.) Computer Security – ESORICS 2022. ESORICS 2022. LNCS, vol. 13556. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17143-7_28
25. Wolsing, K., Wagner, E., Saillard, A., Henze, M.: IPAL: breaking up silos of protocol-dependent and domain-specific industrial intrusion detection systems. In: International Symposium on Research in Attacks, Intrusions and Defenses (RAID). ACM (2022)