ZOE: Content-based Anomaly Detection for Industrial Control Systems

Christian Wressnegger
Institute of System Security
TU Braunschweig

Ansgar Kellner
Institute of System Security
TU Braunschweig

Konrad Rieck
Institute of System Security
TU Braunschweig

***Abstract*—Due its complexity and a multitude of proprietary components, industrial control systems are an immanently difficult field of application for intrusion detection. Proprietary binary protocols and the lack of public specifications have forced the research community to move away from content-based detection to more abstract concepts. In this paper, we show that in contrast to prior belief the content of unknown binary protocols can very well be modeled. ZOE derives *prototype models* that are specific to individual types of messages in order to capture the characteristics of arbitrary binary protocols and enable detecting different forms of attacks as anomalies. In an evaluation based on 6 days of network traffic recorded at a large power plant (1,900 MW) with over 92,000 unique devices, we demonstrate that ZOE improves upon related approaches by up to an order of magnitude in detection performance, but also significantly decreases false positives.**

***Index Terms*—Industrial networks, SCADA, Attack Detection**

I. INTRODUCTION

The protection of critical infrastructures is of utmost importance for society. Industrial facilities, such as power stations and water supply systems, are high-value targets for terrorists and nation-state attackers. The progressing automatization of industrial processes and the interconnection between devices, facilities and control centers significantly increases their attack surface and imposes new challenges for security solutions. A power plant, for instance, consists of a plethora of proprietary software and hardware components from various manufacturers. Many of these components use non-standardized protocols that are specific to manufactures or even to a particular type of device. Consequently, operators of an industrial facility usually do not know about implementation details of the (computer) systems they run. This renders the use of traditional intrusion detection approaches for industrial control systems (ICS) extremely difficult. As a result, the networks in industrial control systems have been increasingly targeted by attacks in the last years [e.g., 7, 15, 33, 47].

Without the availability of protocol specifications that assist in preprocessing network data, an in-depth analysis of communication content is difficult to accomplish. The research community has thus moved towards approaches that model the appearance of network traffic rather than its content [28, 42, 52, 60] or even the underlying physical process itself [24, 29, 61] in order to detect deviations from the expected process states. This more abstract perspective allows for the detection of specific classes of attacks, such as flooding or incremental attacks, but also restricts defense more than necessary. Stuxnet, for instance, has sabotaged the overall production process by making small, infrequent changes to the motor speed of centrifuges [15]. While only subtle changes have been made, these operations happened out of the ordinary and posed anomalies among the usual bus communication with respect to used input values. Thus, given a precise model of normality such anomalies can be detected—both on a process- as well as the network-level.

Constructing these models however is inherently difficult. Industrial facilities are subject to changes in hardware (e.g., sensors, PLCs, etc.) and adaptations of the process itself. On the one hand, using an expert model of the physical process for detection, implies that this model has to be manually updated at every change in order to prevent divergence from reality that in turn may result in loopholes for an attacker. On the other hand, machine learning has been used to automatically learn and update models of communication contents instead. Previous research has demonstrated the effectiveness of anomaly detection for network-based intrusion detection for various fields of application and protocols [63, 65–67, 71, 72]. However, the prevalent use of proprietary binary protocols in industrial networks significantly complicates the use of content-based approaches for the detection of intrusions and often renders existing approaches ineffective [23].

In this paper we attempt to bridge this gap and present ZOE, a framework that effectively and efficiently makes use of content-based anomaly detection for proprietary binary protocols. We show that content models are very well usable for environments that rely on undocumented protocols with high-entropy data. To this end, we introduce the concept of *prototype models*, that is, prototypical representations specific to individual types of messages. These models not only characterize the structure of message types but also the data they typically contain. Moreover, we present a linear-time algorithm for learning and applying these models based on Count-Min Sketches [9], that breaks up the separation of traditional clustering methods and anomaly detection.

For the evaluation of our approach we have recorded roughly 210 GiB of network traffic from two industrial facilities, a large power plant producing about 1,900 MW and a coal mining facility. During 6 days of operation we have gathered data from 92,700 unique devices on *control level and field level* of the facilities. As recordings of attacks in industrial environments are particularly rare, we have additionally developed a tool that automatically generates abnormal network messages based on authentic communication. This enables us to calibrate our detector with varying difficulty. We empirically evaluate our approach based on the six most prominent protocols in the recorded data and show that ZOE not only improves over related approaches by up to an order of magnitude in detection performance, but also significantly decreases false-positives.

In summary we make the following contributions:

- *Prototype Models for Network Messages.* We introduce the concept of prototype models and present a linear-time algorithm for constructing these based on large amounts of network traffic. These models not only characterize the structure of individual message types but also the data they typically contain.
- *Noise-resilient Anomaly Detection.* We demonstrate how protocol models can be used to prune out irrelevant, noisy features that arise from the intermingling of structure and data in binary protocols. This enables us to further enhance the expressiveness of the models and thus allows for robust anomaly detection in environments with high-entropy data.
- *Large-scale evaluation using authentic SCADA data.* We conduct a large-scale evaluation with authentic data from two different industrial facilities for coal mining and power generation. This includes PROFINET IO traffic at the *field level* as well as five entirely undocumented protocols at *control level* with more than 210 GiB of data involving 92,700 unique devices.

The rest of the paper is organized as follows: The problem statement is outlined in Section II. Section III then describes our method ZOE and how prototype models are used for attack detection in proprietary protocols. Section IV describes the data that has been gathered for the evaluation presented in Section V. Related work is discussed in Section VI before Section VII concludes the paper.

II. PROBLEM STATEMENT AND SCOPE

Protecting industrial computer networks, such as ICS and SCADA systems, from attacks is a daunting task. While networks in industrial facilities are much more homogeneous than general-purpose computer networks, they often employ proprietary systems and protocols. In many cases, only few technical details are known about these protocols and even the operators do not have access to the underlying protocol specifications. One reason for this situation is that industrial networks often comprise components of specialized manufacturers, each employing proprietary technology, for example, for controlling turbines and chemical reactors. As a consequence, conventional security techniques, such as intrusion detection systems, are faced with an opaque network environment and few to no information about the exchanged message formats and protocol state machines.

The analysis of communication in industrial networks is further obstructed by the use of binary protocols that rest on compact binary structures for minimizing communication overhead and delay. On the *field level* this frequently is the case due to the limited resources of devices and legacy reasons. Modbus messages, for instance, are limited to 256 bytes as the first implementation has been designed for serial communication over RS485 [39]. Similarly, modern protocols on the *control level* also frequently rely on compact binary structures. For example, five out of the six proprietary protocols considered in our evaluation make use of binary fields and structures.

In the absence of appropriate protocol dissectors and parsers, an analysis of network traffic is only feasible if abstract representations of the exchanged data are developed that are capable of reflecting content and structure in a generic manner. To tackle this problem, we model the communication between two parties in a network as a sequence of incoming and outgoing *binary messages* or more formally application-level data units (ADUs). For stateful transport protocols, such as TCP, these messages can be extracted using regular stream reassembly [11]; for stateless protocols, such as UDP, these messages simply refer to the application-level payloads. In the following, we thus focus on techniques for analyzing binary messages and identifying anomalous content.

An advantage of industrial environments over general-purpose computer networks is that the scope of application is often narrow and clearly defined. The industrial process itself has precisely specified terms of operation and clear expectations with respect to its outcome. This is predestinated for the use of anomaly detection, where a model of normality is inferred using machine learning techniques and deviations from the model are flagged as anomalies. In order to apply machine learning successfully in this context, however, a few things have to be particularly considered [22, 54]:

High costs of error. Wrong classifications are particularly critical in a security context. False-negatives may potentially cause devastating harm to the attacked network, while a high number of false-positives may render an intrusion detection system useless [3]. An effective detector must therefore strive for extremely low false-positive rates and simultaneously detect attacks with high accuracy to keep the overall number of misclassifications and the associated costs low.

Lack of training data. For a learning based detector it is of tremendous importance to operate on sufficiently large training datasets. Anomaly detection attempts to build a model of normality to detect deviations thereof as attacks. This can only succeed if most—preferably all—aspects of the network protocol in question have been considered. Gathering enough training data, however, is a major challenge and not always possible. Next to benign data also malicious samples are quintessential for calibrating and evaluating anomaly detection methods in practice. A fact that is frequently overlooked.

Semantic gap. Anomaly detection does not discriminate between benign and malicious content, but reports deviations from what has been learned as normal content. Any raised alarm hence requires interpretation. To tighten the assumption of anomalies being attacks, periodic retraining of the underlying model is necessary as the notation of normality might change over time. In practice, a linear-time approach for training thus is highly beneficial.

Large variability of input data. Network traffic exhibits great variability in its data and structure. An effective anomaly detection system hence is required to carefully aggregate information from large amounts of data over several days or even weeks to model regular variance and filter out network chaff. For industrial environments this aspect is less severe due to the narrow scope and application of the networks.

III. Attack Detection in Proprietary Protocols

A large body of research on content-based intrusion detection has shown that considering the mere presence of features in network traffic often is superior over counting their occurrences [23, 63, 68, 71]. For example, particular strings might already be indicative to spot network attacks at the application layer. However, ignoring the frequency of features prematurely discards valuable information. While this kind of data may not be mandatory for the pure detection of attacks, we show that it plays a key role in modeling normality and constructing corresponding detection models, in particular, in proprietary network environments.

Based on this observation, we develop a content-based anomaly detector, Zoe, that in contrast to previous work is capable of robustly handling binary and text-based protocols likewise, without requiring any knowledge of the underlying specification. To this end, we learn a model of normality from observed network traffic and detect attacks as deviations thereof. The important difference to related approaches is the use of adequate ways of modeling unknown protocols. To achieve this goal we rely on two key components, that seamlessly intertwine:

A. *Building Prototype Models.* For constructing a model of normality we automatically partition network traffic into k groups of messages with similar content, thereby approximating states of the underlying protocol. This procedure is designed to not only separate message types but also, to derive one prototype model per message type in the process, which makes a classical separation of clustering and subsequent learning of content models unnecessary.

B. *Reducing Noise.* One of the main obstacles for analyzing unknown protocols is "noise", that is, seemingly random data that hinders inferring suitable content models [see 23]. We address this problem by analyzing the occurrences of features in each prototype model and carefully filtering rare features using a frequency threshold t.

These two components form the basis of our detector, which thus is parametrized by the number of message groups k and the frequency threshold t:

$$\text{Zoe}(k, t)$$

The usage of one building block without the other can be denoted as $\text{Zoe}(1, *)$ for a detector using one global content model rather than individual prototype models but different thresholds for noise-reduction, and $\text{Zoe}(*, 0)$ for the use of prototype models that however do not filter noise.

A. Building Prototype Models for Network Messages

The messages monitored in an industrial network can be represented as strings of variable length. While this representation is ideal for conventional signature-based detection, for building protocol models we however require a more structured representation of the data. We thus map each message m monitored in the network to a corresponding feature vector $\mathbf{x} = \phi(m)$: We extract all substrings of length n—so called n-grams—from a message m and record their occurrences. Each substring is associated with one dimension of the feature space, such that a message m can be expressed as a vector of substring occurrences. Formally, this map is defined as follows

$$\phi\colon m \to \big(\phi_s(m)\big)_{s \in S} \quad \text{with} \quad \phi_s(m) = \operatorname{occ}(s, m)$$

where the set S denotes all possible substrings of length n and the function $\operatorname{occ}(s, m)$ represents the occurrence of the substring s in the input message m. This can be implemented as the frequency, the probability or a binary flag for a feature's presence. Using this mapping, we can translate a set of messages $\{m_1, \ldots, m_N\}$ to a set of vectors

$$X = \{\mathbf{x}_1, \ldots \mathbf{x}_N\} \quad \text{with} \quad \mathbf{x}_i = \phi(m_i).$$

Depending on the length n, the vector space can be high dimensional, as the number of considered substrings grows exponentially with n. Fortunately, the resulting vectors are very sparse and thus efficient data structures for handling sparse data can be applied to operate in this vector space.

Based on this representation we can now proceed to learn prototype models as a first cornerstone of our detector to cope with high-entropy data in binary protocols. To learn models per message type we build on methods from the field of clustering. Unfortunately, clustering is a rather expensive task and many algorithms are not suited for efficiently processing large amounts of data. With Zoe we aim at an integrated solution that breaks up the separation of clustering and subsequent learning of content models. We thus build upon a linear-time algorithm for approximating clusterings [1] that we have tailored to network traffic analysis such that we can build prototype models on-the-fly.

We first transform our network messages m to feature vectors $\mathbf{x}$ as describe before, to build the input dataset X. Then, k samples are drawn from the input data to initialize clusters $C_1, \ldots, C_k$. For each following sample $\mathbf{x} \in X$ we measure the similarity to each cluster and assign it to the cluster C_j that has the closest proximity

$$j = \underset{i \in [1,k]}{\arg\max}\ \operatorname{prox}(\mathbf{x}, C_i)$$

The overall similarity is calculated based on the input sample $\mathbf{x}$ and all samples that belong to a cluster C:

$$\operatorname{prox}\colon \mathbf{x}, C \rightarrow \frac{1}{|C|} \sum_{\mathbf{y} \in C} \operatorname{sim}(\mathbf{x}, \mathbf{y})$$

The measure sim for two input samples may be either approximated by the dot product $\widetilde{\operatorname{sim}}\colon \mathbf{x}, \mathbf{y} \rightarrow \mathbf{x} \cdot \mathbf{y} = \sum_{i=0}^{n} x_i y_i$ or defined by the cosine similarity based on the l^2-norm $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=0}^{n} x_i^2}$, that is,

$$\operatorname{sim}\colon \mathbf{x}, \mathbf{y} \rightarrow \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2} = \cos(\theta)$$

where θ denotes the angle between the vectors $\mathbf{x}$ and $\mathbf{y}$. While this requires slightly more effort to realize a linear-time implementation, the normalized angle has the advantage of actually being a formal distance metric.

For a sufficiently large input set X the algorithm above approximates a precise clustering with high probability [1]. This algorithmic requirement however also demands an especially efficient way of handling the sets of messages as clusters. We hence make the following two optimizations in our implementation: First, we store counts of messages and their substrings rather than the messages themselves in order to save valuable working memory. For each cluster C_i we only maintain the total number of all samples $|C_i|$ contained in the cluster and a vector of cumulative counts as *prototype models*:

$$\mathcal{P}_{\mathbf{i}} = \sum_{\mathbf{y} \in C_i} \mathbf{y}$$

This suffices to compute the similarity between messages and clusters as defined before, and can be used for efficient anomaly detection in further follow (cf. Section III-B).

Second, due to the large amounts of network traffic we operate on, storing and keeping track of substring counts already poses a considerable challenge. Retaining exact counts simply is not feasible as it requires to store all substrings (or at least hashes thereof). We thus revert to probabilistic counting of substrings. In particular, we make use of *Count-Min Sketches* [9], a probabilistic data structure that is closely related to Bloom filters [4] but additionally allows for counting occurrences rather than answering membership queries only.

A Count-Min Sketch is defined as a $w \times d$ two-dimensional array of numeric items of arbitrary size and precision, and d hash functions h_i that map input strings to numeric values in the interval $[0, w-1]$. To store a particular key-value pair or increment the value for a key in the sketch, each hash function is at first applied to the key, for instance a substring $s \in S$ appearing in message m. As depicted in Figure 1, the resulting value is then used as position in the corresponding row. At these offsets the stored numeric value is incremented by the provided value v. Retrieving the value for a key works analogous: The hash functions are applied to the key in order to determine the numeric values associated to it. The minimum of these values then recites the approximate true value—in our case the approximate count of feature/substring s.

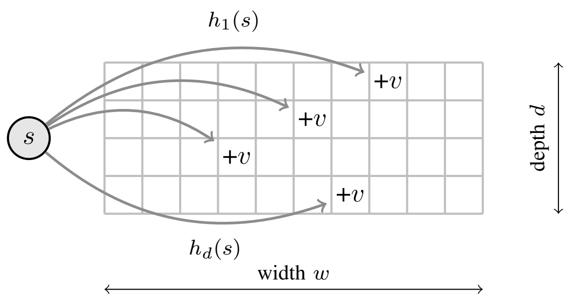


Fig. 1. Schematic depiction of a Count-Min Sketch [9]. The substring to be added is denoted as s and is processed by d (the depth of the sketch) hash functions h_i to determine the position $p_i = h_i(s)$ with $i \in [1,d], p \in [0,w)$ at which value v is added.

By construction, the approximated count $\hat{c}$ retrieved from a Count-Min Sketch is always larger or equal to the true count c, meaning that the data structure will never underestimate a stored value:

$$c \leq \hat{c} \quad \forall c \in \mathbf{c}$$

where $\mathbf{c}$ is the vector of all values stored in the Count-Min Sketch. Furthermore, for a width $w = \lceil \frac{e}{\varepsilon} \rceil$ and a depth $d = \lceil \ln \frac{1}{\delta} \rceil$ it is guaranteed that the difference between the true and approximated value is at most $\varepsilon \|\mathbf{c}\|_1$ with a probability p of at least $1 - \delta$ [9]

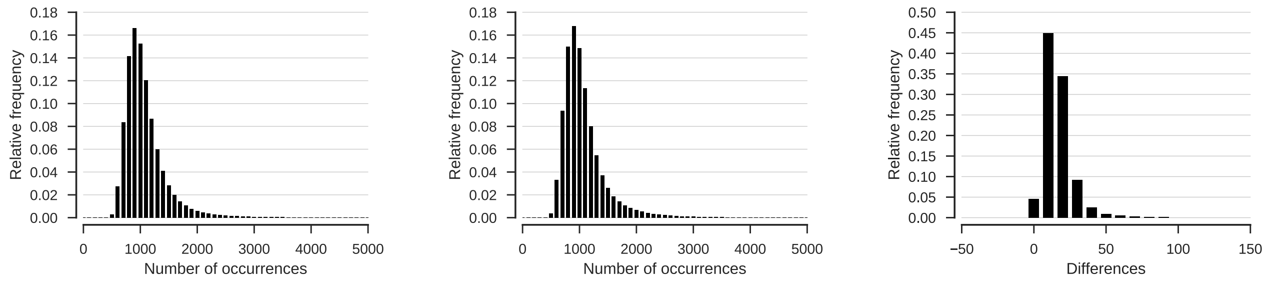
$$\hat{c}_i \leq c_i + \varepsilon \|\mathbf{c}\|_1$$

In other words, the estimate of the Count-Min Sketch is correct within ε times the number of items stored in the data structure with a probability of p. Figures 2(a) and 2(b) show the distribution of the relative frequency of occurrences of strings in our evaluation data, once counted exactly and once probabilistically using a Count-Min Sketch with $\varepsilon = 0.0001$ and $\delta = 0.01$, resulting in a width $w = 27{,}183$ and the use of $d = 7$ hash functions. Figure 2(c) shows the relative frequency of the difference in value of these counts. Most values differ by roughly 10 to 20 occurrences and none, as stated above, is lower than the true value.

B. Noise-resilient Anomaly Detection

As a second building block for reliably detecting attacks in proprietary network protocols, we propose an extension to content-based anomaly detection made possible by prototype models introduced in the previous section. Language models, such as n-grams [e.g., 56, 66, 68], have frequently been used for attack detection and have proven impressively effective for *text-based* data [e.g., 45, 66–68]. For *binary* and *high-entropy* data, however, such models are considered mostly impractical by the research community so far [23] and have been widely discarded for application in industrial networks.

The main reason why language models perform worse in this environment is founded in the density of the message data.



(a) Exact counting (b) Probabilistic counting (c) Exact vs probabilistic counting

Fig. 2. Histograms of 3-gram occurrences in network traffic: a) exact counting, b) probabilistic counting, c) the difference between both approaches.

The density is defined as the ratio of the number of unique occurrences of a feature/substring s to the total number of all possible elements S [71]. This ratio significantly influences the quality of the resulting model. In case of high entropy, as induced by binary network protocols, a model gets so "packed" that it becomes difficult to differentiate between two classes (benign and malicious messages) and renders mere binary embedding impractical [23, 68]. Count embeddings, on the other hand, have been shown to be less effective when used as a drop-in replacement in an otherwise identical setting [23, 63, 68, 71].

Rather than discarding this information altogether, we use the number of occurrences to filter relevant from irrelevant information. Note, that we record the frequency f of samples in which features/substrings $s \in S$ occur rather than the total count of s in the complete data set. Setting f in relation to the total number of samples N formally yields the well-established *document frequency* measure $df = \frac{f}{N}$. The prototype models $\mathcal{P}_\mathrm{i}$ that we have established in the previous section represent exactly these feature frequencies across training samples iff function $\mathrm{occ}(a, m)$ is defined to report the existence of substring s in input message m. Detection schemes based on binary embeddings can thus be directly derived from prototype models $\mathcal{P}_\mathrm{i}$ *without additional training*.

By introducing a threshold t we now prune features that occur in less than t input samples that have been associated with a cluster and thereby effectively discard noise from the training data. Models M_i can hence be interpreted as sets of features that are considered for detection and together form the overall content model used by ZOE:

$$\mathcal{M} = \{M_1, \dots, M_k\} \text{ with } M_i = \{s \in S \mid \mathcal{P}_{i,a} \geq t\}$$

This allows us to revert to detection using the (implicit) binary embedding based on the remaining, most relevant features only. This scheme offers two main advantages: First, it allows to reduce the set of benign features to those that appear more than t times, and thus significantly reduces the size of the model as features associated with a value of 0 are not explicitly stored. Furthermore, this limits an attacker's reach of play when mimicking benign messages (Section V-E). Second, for production use and to improve runtime performance of the final detector this binarized Count-Min Sketch can be transformed into a compact Bloom filter that offers a higher accuracy than established methods [68] with the same memory footprint (Section V).

In order to determine the overall detection using k models ZOE considers the score of the model with the highest resemblance to the message m in question. With a scoring function d that yields low values for known/benign messages and high values for anomalies, this formally translates to choosing the minimum score of k models:

$$\mathrm{score}\colon m, \mathcal{M} \to \min_i \mathrm{d}(m, M_i)$$

In accordance to d and using an overall threshold T, a message is considered malicious for $\mathrm{score}(m, \mathcal{M}) \geq T$ and benign otherwise. Different schemes on how to choose scoring function d and how to adjust and interpret the detection threshold are discussed in the subsequent section.

C. Adjusting the Detector

One particularly effective way of evaluating a message m is to determine the ratio of known or unknown features to the total number l of features in the message:

$$\mathrm{d}_1\colon m, M \to 1 - \frac{1}{l} \sum_{s \in M} \mathrm{occ}(s, m)$$

While this distance has been shown to be effective in a number of applications [23, 49, 68, 71] interpreting the threshold for such a scoring function is rather difficult in practice. Different measures based on the number of bytes covered by the model, for instance, however often are not performant enough to match up. The benefit of the latter is that the operator of the intrusion detection system is able to a) more naturally specify the necessary threshold T as the number of previously unobserved bytes a message may contain before a message/packet is considered to be part of an attack, and analogous b) intuitively interpret the resulting scores as the portion of unknown content in bytes.

$$\mathrm{d}_2\colon m, M \to \mathrm{cov}(m, M)$$

The function $\mathrm{cov}(m, M)$ returns the number of bytes in m covered by model M. Normalizing to the length of the

TABLE I
DATASET AND PROTOCOLS USED FOR THE EVALUATION OF ZOE.

Protocol	Generation					Mining
	P1 2000	P2 2069	P3 4241	P4 2010	P5 2070	P6 PROFINET IO
TCP	●	●	●			(●)
UDP				●	●	
Binary		●	●	●	●	●
Text	●		●	●	●	
Size	2,542 MiB	219 MiB	18,506 MiB	2,506 MiB	1.6 MiB	1,878 MiB
Count	7,032,323 msgs	310,205 msgs	13,046,151 msgs	16,024,175 packets	33,329 packets	13,957,589 packets

message is explicitly avoided. To a certain extent this takes away the flexibility of the model but most importantly also significantly raises the bar for an attacker to perform mimicry attacks (cf. Section V-E).

IV. DATASETS

For evaluating our approach under realistic conditions, we have partnered with a large European energy producer. In particular, we have collected roughly 210 GiB of raw network data during 6 days at a large power plant (total 1,900 MW), and the operation of a coal mining facility. At the power plant we have recorded network traffic at the *control level* of a power unit producing 500 MW. Our recording period covers a ramp-up phase as well as normal operation of the unit. From this we have extracted the five most prominent protocols, all of which are proprietary and publically undocumented[1]. Through manual analysis we however were able to attribute these to a large plant manufacturer. In order to provide a comprehensive study on protocols that occur in industrial networks the data recording at the coal mining facility targets PROFINET IO at the *field level*. In total we have recorded communication between 92,700 unique devices in an authentic production environment. Table I summarizes the gathered data.

Protocols P1–P3 build on TCP while P4 & P5 use UDP for direct communication. PROFINET IO, on the other hand, is situated somewhat differently: While TCP/IP is used for the parameterization and configuration, real-time messages are exchanged on a separate channel that does not use the Internet protocol at all. Although PROFINET IO traffic can be easily parsed and therefore reliably filtered, there are no protocol parsers publicly available for the remaining traffic from the power plant. For these protocols we hence resort to filtering the network traffic based on IP ports, but explicitly consider relations between communicating entities to sanitize the data. For TCP traffic we additionally reassemble network streams, such that we are able to evaluate ZOE based on complete “messages” (approximated as consecutive, unidirectional traffic) for P1–P3, datagrams for P4 & P5, and PROFINET IO packets. All in all, this gives us 46.7 GiB of raw data for our evaluation.

Furthermore, these different protocols show highly diverse structure. While protocols P1 and P2 seem to exclusively use either text-based or binary-based data for their communication, protocols P3–P5 use a mixture of both. The latter appear to mainly consist of binary structures that additionally transmit string-based (printable) data. As these strings exhibit strong structure that may be interpreted as another protocol on top of the base protocol we assign these to both sub-groups. PROFINET IO then again is strictly based on binary data. Additionally, server and client communication often vary significantly for these protocols. To account for this difference in composition and structure we thus split the individual protocol subsets in *incoming* and *outgoing* traffic and analyze these individually in our evaluation. UDP and PROFINET IO traffic is not effected by this pre-processing step.

A. Attack Datasets

Effective anomaly detection can only succeed with a carefully chosen parametrization of the detector. This requires benign traffic for building the content model, but also attack samples to validate the chosen settings and estimate the expected detection performance [54]. As recordings of such attacks in industrial environments and for proprietary protocols in particular are naturally rare, we have developed a tool for the automatic generation of network attacks against unknown protocols. This tool provides us with a total number of 3,899 unique attacks for each protocol P1–P6, where we limit the payloads to 256 bytes in size. By restricting the length of the attack strings, we ensure that these do not outweigh the benign content and hence better blend in.

In particular, the tool mimics a protocol based on observed network traffic as close as possible and injects attack strings, that range from program code (e.g., shellcodes, ROP chains, PLC instructions), over scripts fragments (e.g., Perl scripts) to random data, at positions containing variable input. To this end, we first group similar network messages using clustering and then derive generic rules that describe the structure of the messages in each cluster (Section IV-A1). Second, we generate attack strings with different encodings and obfuscations to populate variable fields of the derived rules (Section IV-A2).

1) Inferring Protocol Rules: For industrial facilities we usually do not have the specification of the used network protocols on hand and neither does the operator, as vendors normally do not share product details. Consequently, we are required to automatically infer protocol models based on

[1]Consequently, publicly available tools such as Wireshark are not capable of parsing these protocol beyond the TCP/UDP packet structure.

network traffic only [10, 21, 32, 34]. To this end we build upon work by Krueger et al. [32] and Gascon et al. [21] to derive rules for individual messages.

In a first step, we identify messages that have the same structures using off-the-shelf k-means clustering. Similarly to the method described in Section III-A network messages cannot be directly used at this point, but are embedded into vector space first. We thus again operate on an input set $X = \{\mathbf{x}_1, \dots \mathbf{x}_N\}$ yield by the feature map ϕ with a binary embedding of substrings $s \in S$. However, as inferring protocol models is much more involved than simply deriving states, additional statistical tests are applied to filter relevant from irrelevant substrings/features and to avoid an overly populated vector space that might hinder the clustering process. We hence subdivide the feature space in *constant*, *mutable*, and *volatile* parts by applying a binomial test to each feature [31]. A value close to 0 indicates volatile features while a frequency of 1 refers to constant features. Neither volatile nor constant features are particular valuable for discriminating messages in a protocol—think of a protocol's magic values or nonces. We hence reject all features that do not meet a statistical significance level of $\alpha = 0.05$ [25] before clustering the inputs.

Subsequently, we derive rules that describe all messages in a cluster which can then be used to generate new messages that comply with the format of that cluster. To do so we consider the original messages in each cluster, rather than the reduced representation in feature space and pair-wise align these using an extended version of the Needleman-Wunsch algorithm [40]. The resulting rules consist of *fixed bytes* that appear at the same position in each message of the cluster and *variable fields* that may contain different byte sequences from message to message. Table II shows an example for a set of simple messages found in our datasets.

TABLE II
RULE INFERENCE FOR A SET OF SIMPLE MESSAGES.

Message 1	`\x00\x1a GetVal "N2HAJ22CT000 XH24"\xff`
Message 2	`\x00\x1a GetVal "N2HAJ22DT500 XR01"\x00`
⋮	⋮
Message n	`\x00\x1a GetVal "N2HAJ31AA100 XB02"\xff`
Rule	`\x00\x1a GetVal "N2HAJ` $\boxed{\quad\quad}$ `00 X` $\boxed{\quad}$ `"` $\boxed{\quad\quad}$

It is clear to see that our approach is not capable of learning the exact protocol specification but approximates it based on the network traffic on hand. The depicted cluster contains messages that apparently transmit a command string `GetVal` to retrieve sensor values from a particular device. Furthermore, it contains the length of the transmitted string as 2 byte integer in the front and another unspecified flag (presumably a high and low value) at the end. Within this cluster only individual parts of the device identifier and the binary flag at the end change such that the remaining parts are considered constant by the algorithm. For populating a valid network message with attack payloads this however is sufficient.

2) Generation of Attack Messages: With the inferred rules as detailed descriptions of protocol messages at our disposal we can now produce authentic network traffic that contains arbitrary attack payloads. For a wide range of different attacks we query Metasploit for payloads and encoders—simple obfuscations that, for instance, encrypt scripts with a simple `xor` operation, or map program code to printable characters. To generate attack messages we proceed as follows: For each attack string we randomly choose an existing network message. The corresponding rule then allows us to replace variable fields within this message at will. For our experiments we choose exactly one at random and inject the attack payload. The remaining (constant and variable) fields remain unchanged. Figure 3 depicts this simple scheme.

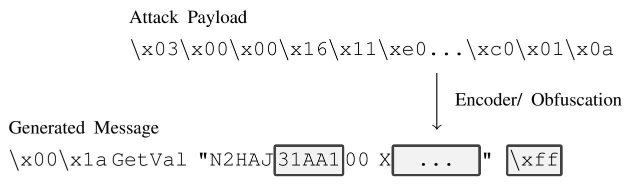

Attack Payload
\x03\x00\x00\x16\x11\xe0...\xc0\x01\x0a
Encoder/ Obfuscation
Generated Message
\x00\x1a GetVal "N2HAJ 31AA1 00 X ... " \xff

Fig. 3. Schematic depiction of the generation of network attacks.

V. Evaluation

We empirically evaluate Zoe based on the six industrial protocols described in the previous section and conduct a number of experiments that examine the different aspects of our method: First, we demonstrate the overall detection performance of our method as proposed in the paper (Section V-A). Second, the influence of Zoe's de-noising capabilities as well as the impact of message-specific prototype models is studied (Section V-B & V-C). Third, we compare our method with related approaches (Section V-D) before, as a final experiment, we investigate the feasibility of evasion attacks in the form of polymorphic blending attacks against Zoe (Section V-E).

In the course of this evaluation we describe detection performances with the aid of the *receiver operator characteristics (ROC)* and corresponding ROC curves. These curves plot the true-positive rate over the false-positive rate of a detector for different thresholds [6, 16]. Additionally, we use the area under the ROC curve (AUC) as a single continuous measure for the detection performance that yields a minimal and maximal value of 0.0 and 1.0, respectively.

Figure 4 illustrates a ROC curve that shows the full scale of false-positives in the interval $[0.0, 1.0]$ on the x-axis or $0\,\%$ to $100\,\%$. The AUC can hence be interpreted as a measure of how steep the curve increase towards higher true-positive rates. As measuring the AUC for the full range often is of little expressiveness (performances for low false-positives are poorly represented) we use the *bounded AUC*. That is the area under the ROC curve up to a threshold b of false-positives and normalized to that value: AUC(b). For the field of attack detection it is particular important to push forward detection with few false-positives.

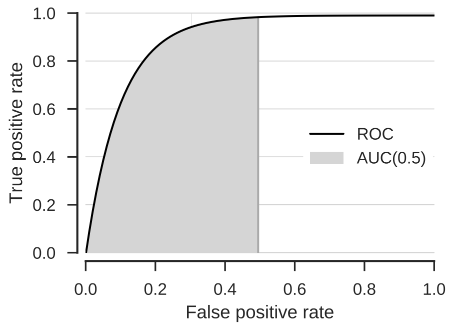


Fig. 4. Exemplary ROC curve with the bounded AUC (area under the curve) for a false-positive rate of 0.5 or $50\,\%$, respectively. Setting this bound allows to zoom in on the critical range of low false-positives that is particular important for an detector.

For training and testing individual detectors we create strictly separated datasets that do not overlap. $75\,\%$ randomly chosen samples are used as *known data* for training and the remaining $25\,\%$ as *unknown data* for testing. This partitioning is applied to benign and malicious samples likewise, and is repeated for 10 experiments which are averaged to determine the overall detection performance. The parameters for the detector are chosen solely based on the results obtained on training data and only then, this configuration is used to evaluate the performance on the testing dataset.

A. *Overall Detection Performance of* ZOE

We begin with demonstrating our method's detection performance for the six industrial protocols we have collected. Figure 5 shows the results for our detector parametrizes with different de-noising thresholds and numbers of prototype models using 5-grams. The y-axis shows the detection performance as AUC limited to different thresholds of false-positives, over the individual protocols on the x-axis. ZOE performs very well across different protocols and detection performance only differs in nuances. The parametrization of the detection however is crucial as we demonstrate in the following sections. For protocol P3 (outgoing), for instance, our experiments show that at least 4 different prototype models are needed to enable good detection. Subsequently, we thus inspect the influence of the individual components of ZOE in detail and use protocol P3 as recurring example.

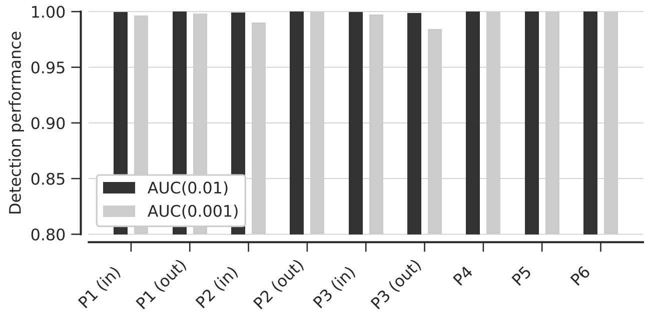


Fig. 5. Detection performance of ZOE for the protocols P1 to P6.

B. *De-noising Content Models*

Next, we evaluate the impact of different thresholds t used with ZOE to demonstrate its de-noising capabilities. We hence parametrize ZOE$(1, t)$ with thresholds $t \in [20, 100]$ with a granularity of 5. Figure 6 shows the results for protocol P3. At a threshold $t = 35$ the detector reaches its peak performance with an AUC(0.0001) of 0.8 (light gray line) and a true-positive rate of 0.978, meaning that $97.8\,\%$ of the attack patterns are detected with at most 1 false positive out of $10{,}000$ network messages. In comparison to no de-noising ($t = 0$) we record an tremendous improvement that clearly shows that pre-filtering the features used for detection is of the essence for effective attack detection in binary protocols.

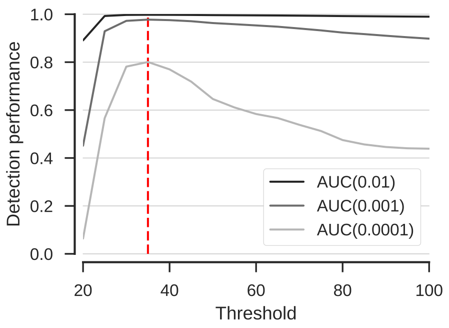


Fig. 6. Detection performance of ZOE$(1, t)$ for varying thresholds t. The dashed line indicates the threshold yielding the highest detection performance ($t = 35$).

Moreover, it is interesting to see that the detection performance shoots up at thresholds of 20 to 25, peaks at $t = 35$ and flattens out towards higher values. This underlines the importance of a thorough evaluation with the aid of a comprehensive set of attack samples.

C. *Message-specific Prototype Models*

In this section we bring together the two key components of ZOE and study the influence of using multiple prototype models on the detection performance. Furthermore, we explicitly highlight the additional improvement in the eminently important range of low false-positive rates.

To this end, we train our detector for different numbers of prototype models k and thresholds t in order to calibrate the detector as described in the previous section. The best configuration with both components enabled, ZOE$(4, 50)$, is shown as a ROC curve in Figure 7 in relation the best detector without prototype models, ZOE$(1, 35)$. Note, that the x-axis (false-positive rates) shows a logarithmic scale to better emphasize the improvement, that the detection specific to message types entails. Such enhancements in detection performance are of great importance for the application of the detector in production and limits the costs of false-alarms.

D. *Comparison with Related Approaches*

Ultimately, we conduct a comparison of ZOE to Anagram [68], a content-based approach using higher-order n-grams ($n \geq 3$). While it originally has been designed for

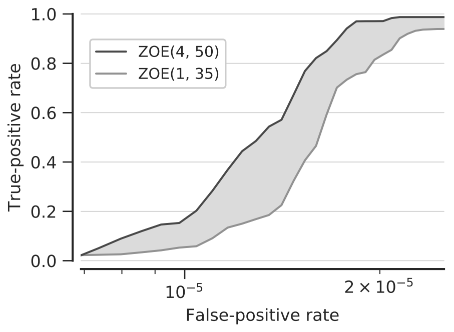


Fig. 7. Detection performance of two configurations of ZOE: First, *a global content-model* with $(1, 35)$ and second, *multiple prototype models* with $(4, 50)$. The latter clearly improves upon the other in the critical region of low false-positive rates.

detecting attacks in HTTP traffic it has proven effective for various other protocols as well [23, 71]. Previous research has shown that it even works well for certain binary protocols with a limited set of different message formats and simple structure such as Modbus [23].

Anagram is similar to the most basic configuration of our detector, ZOE$(1, 0)$, that neither make use of prototype models $k = 1$ nor de-noising of the content models $t = 0$. In order to maintain comparability in this experiment we make use of filters/sketches that have approximately the same number of items for storing content features. We hence choose $\varepsilon = 0.0000006$ and $p = 0.99$ for Count-Min Sketches in order to match the 2^{25} items large Bloom filter we use for Anagram in this experiment. We also make use of d_1 as distance measure for a direct comparison and choose ZOE's best parametrization with $k = 4$ and $t = 50$.

TABLE III
DETECTION PERFORMANCE ZOE AND ANAGRAM FOR PROTOCOL P3.

	Detection performance		
Method	AUC(0.01)	AUC(0.001)	AUC(0.0001)
ZOE	0.9984	0.9844	0.8463
Anagram	0.1515	0.0408	0.0002

Table III shows the detection performance of both as AUC bounded to different thresholds of false-positives for protocol P3. While ZOE yields high performance values down to a false-positive rates of 0.0001, Anagram is not able to compete and only achieves an AUC(0.01) of 0.15, which even reduces to 0.04 for AUC(0.001). This is founded in the tight connection of structure and data found in binary protocols, which conventional content-based anomaly detection is not designed to operate on.

In the ROC curve shown in Figure 8 the difference becomes even more apparent. Again, a logarithmic scale has been chosen to highlight the importance of low false-positive rates for intrusion detection. ZOE yields a true-positive rate of 0.971 for as few as $0.002\,\%$ false-positives, meaning that, $97.1\,\%$ of the attacks are correctly detected with only 2 false-alarms out of $100{,}000$ messages. Anagram, on the other hand, only detects $16.5\,\%$ with an $500\times$ higher false-positive rate of $1\,\%$. These results show that ZOE not only outperforms Anagram by an order of magnitude in sheer detection performance, but also in terms of false-positive rates.

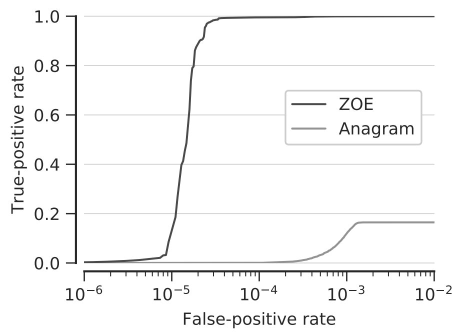


Fig. 8. Detection performance of ZOE and Anagram side-by-side as ROC curves on a logarithmic scale to better emphasize the extremely small false-positive rates achieved by our method.

E. Evasion

In order to bypass content-based anomaly detection an adversary might attempt to mimic benign content and slip by attacks. We thus study the effects of a particular kind of evasion attack—polymorphic blending—on our detector.

Mutation and transformation attacks have a long-standing history for evading intrusion detection systems on different levels [e.g., 18, 19, 44, 51, 55, 57, 62]. Polymorphic blending attacks are a particular effective tool against content-based intrusion detection [19]. Although generating these has been proven to be NP-hard [18], attackers can resort to heuristics to provide good approximations, for instance, using iterative hill climbing. This method from mathematical optimization starts off with an initial solution (the original attack) and iteratively improves it according to a target function (the scoring function of the detector).

Two aspects are key to the success of the attack: First, the use of lower-order n-grams and second, a rich and extensive model containing large amounts of benign features that can be utilized by the attacker. The latter can also be expressed by the density and variability of the training data set [71] that directly influence the richness of the model. Note, that using higher-order n-grams ($n \geq 3$) also increases the complexity of the model due to rare but apparently benign features. With ZOE we address both aspects by a) using substrings of length $n = 5$ and b) pruning rare features, thus limiting the reach of play of an attacker.

In this final experiment we generate $32{,}640$ blended attack instances per epoch over a total of 100 epochs. Additionally, after each iteration we select the 10 best performing instances for further mutations. This results in $3{,}234{,}624$ variations that are tested for each attack. Figure 9 shows the results of a single experiment for a number of different thresholds of ZOE. The gray-scale level decreases with higher thresholds from black ($t = 35$) to light gray ($t = 50$). The higher the detection score to which the individual curves converge to, the more robust the detector is against polymorphic blending attacks. This clearly shows that narrowing down the set of features that are used in the content model (increasing the threshold) is not only beneficial for the detection performance itself but also for improving the resistance against mutation attacks.

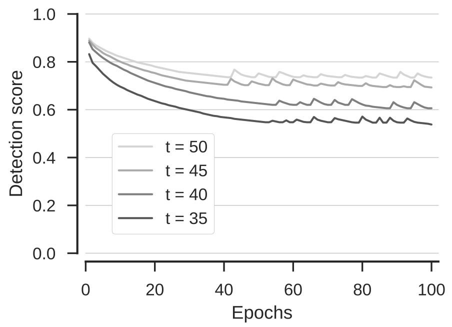


Fig. 9. Polymorphic blending attacks based on 3,234,624 mutations over 100 epochs for different thresholds t used with ZOE.

The ripples observed in the second half of the curves indicate the point in time when the algorithm has detected a potential local minimum and attempts to escape by performing a small number of random mutation at once.

VI. RELATED WORK

Anomaly detection has a long standing history in computer security research and in the scope of network-based intrusion detection in particular [e.g., 26, 45, 48, 65–68]. The uprise of SCADA security has again fostered the development of new methods and has extended the scope such that we differentiate between the following, orthogonal strains of research: A) content-based attack detection, B) detection based on network characteristics, and C) detection by modeling physical processes. Subsequently, we review related work with respect to these different directions with a special focus on industrial control systems. Due to the proximity to our approach we however also include works on content-based anomaly detection in general-purpose networks.

A. Attack detection based on Content

A large body of research deals with the in-depth analysis of protocol contents. This includes the identification [13], analysis [41, 70] and reverse-engineering [8, 35] of protocols, but also the detection of shellcodes in network streams [46, 53]. ZOE in contrast strives for a less specific approximation of protocols tailored to the purpose of attack detection, and aims at a wider range of attacks than shellcodes and abnormal behavior in general.

For this purpose language models such as n-grams have been proven to be very effective [45, 48, 56, 66–68]. Early approaches [30, 66] build distributions of byte frequency as a notion of normality to detect attacks as deviation of these histograms. PAYL [67] extends these to the use of 2-grams, before Wang et al. [68] propose Anagram and establish the use of higher-order n-grams ($n \geq 3$). Similar to our approach Anagram stores a sparse vector representation of content data in a probabilistic data structure. Recently, these data structures have also been used for protocol-specific anomaly detection (in combination with LTSM networks) for SCADA systems [17]. Rieck and Laskov [48] address the use of higher-order n-grams with a representation as Trie that allows for the efficient computation of distances of such vector representations. Spectogram [56], on the other hand, extracts n-grams from HTTP request and models these as a mixture of Markov chains and thus avoids storing observed features altogether. Moreover, it also employs a variant of clustering to control the number of Markov models. While similar in spirit to ZOE, both approaches operate on an entirely different scale.

Wressnegger et al. [71] and Hadžiosmanović et al. [23] provide a more general overview of content-based detection based on n-grams. The latter however focuses on industrial control systems and inspects the suitability of various detection methods such as POSEIDON [5] and Anagram [68]. In concept similar to PAYL, Düssel et al. [14] presents an anomaly detection system based on n-grams using distance metrics for industrial networks. This certainly shares the motivation with ZOE but does not pursue any advanced detection strategies to cope with proprietary binary protocols found in large industrial facilities.

B. Attack Detection based on Network Characteristics

Alternatively to evaluating the content of network packets, researchers have considered sequences of network packets and the relationships of the communicating devices. These approaches operate on a complementary level of network traffic compared to ZOE. Yang et al. [73] present an intrusion detection system that uses an auto-associative kernel regression model coupled with the statistical probability ratio test. Schuster et al. [52] apply a one-class SVM on a number of traces from real-world industrial traffic from different industrial control systems. Koutsandria et al. [28] and Parvania et al. [42] propose to extend intrusion detection systems for SCADA systems by combining traditional signature-based approaches and communication rules, while considering physical limits of the involved devices. Fovino et al. [20] propose a new state-based intrusion detection system for SCADA systems that combines traditional, signature-based techniques with a state-analysis technique. Udd et al. [60] extend the network security platform Bro [59] to support a particular SCADA protocol.

C. Attack Detection by Modeling Physical Processes

Several researchers attempt to model the physical process of industrial facilities which again is orthogonal to our approach. Teixeira et al. [58], Alajlouni and Rao [2], and Vukovic and Dán [64], for instance, model system state estimators and analyze their security properties and detect emerging anomalies. Luchs and Doerr [36] have recently presented an anomaly detection scheme that makes use of envelope escalation for sensor readings. Hadžiosmanović et al. [24] model the semantics of process variables while Mo et al. [38] propose a model-based technique to detect integrity attacks on the sensors of cyber-physical system.

Another strain of research formally describes the underlying process. Pasqualetti et al. [43] propose a mathematical framework for modeling cyber-physical systems and attacks. Wang et al. [69] utilizes a detection scheme based on relation-graphs to detect stealthy false-data injection attacks, while Miao et al. [37] use linear combinations of coding sensor outputs to detect those attacks. Do et al. [12] formulate the attack problem as transient changes in stochastic-dynamical systems involving unknown system states. Rocchetto and Tippenhauer [50], on the other hand, use formal modeling to discover potential attacks on cyber-physical systems.

Other model-based approaches deal with the detection of manipulated physical data—frequently using clustering techniques: Krotofil et al. [29] propose a process-aware approach to detect sensor signal manipulations using the correlation entropy in clusters of related sensors. Kiss et al. [27] detect cyber attacks targeting measurements using a Gaussian mixture model to cluster sensor measurements. Urbina et al. [61] study the physics-based detection of attacks in control systems and develop an adaptive adversary model as well as a new metric for measuring the impact of stealthy attacks.

VII. Conclusion

While the use of n-grams has been proven very effective for content-based anomaly detection of *text-based* network protocols [23, 65–67, 71], for *binary-based* protocols these however are considered mostly impractical [23]. For industrial environments and SCADA systems the research community has thus moved towards alternative schemes, such as modelling the appearance of network traffic or the underlying physical process.

In contrast to prior believe we show that content-based anomaly detection very well is applicable to protocols with high-entropy data. We observe that the frequency values of features carry valuable information for modeling normality and constructing detection models—especially for proprietary binary protocol. The combination of learning message-type specific prototype models and de-noising these enables effective attack detection with particularly few false-positives.

In an extensive evaluation using $210\,\mathrm{GiB}$ of network traffic from two industrial facilities, we show that Zoe is able to significantly improve detection performance compared to related approaches. As an example, our method detects $97.1\,\%$ of the attacks in our dataset with as few as 2 false-alarms out of 100,000 messages, while Anagram only yields $16.5\,\%$ with an $500\times$ higher false-positive rate of $1\,\%$. Moreover, we study the influence of polymophic blending attacks on our detector and show that Zoe's de-noising functionality effectively limits an attacker's reach of play and improves the resistance against this type of mutation attacks.

Acknowledgment

The authors gratefully acknowledge funding from the German Federal Ministry of Education and Research (BMBF) under the project INDI (FKZ 16KIS0154K).

References

[1] C. Aggarwal. A framework for clustering massive-domain data streams. In *Proc. of International Conference on Data Engineering (ICDE)*, 2009.

[2] S. Alajlouni and V. Rao. Anomaly detection in liquid pipelines using modeling, co-simulation and dynamical estimation. In *Proc. of International Conference on Critical Infrastructure Protection (ICCIP)*, 2013.

[3] S. Axelsson. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 1999.

[4] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communication of the ACM*, 13(7), 1970.

[5] D. Bolzoni, S. Etalle, and P. Hartel. POSEIDON: A 2-tier anomaly-based network intrusion detection system. In *Proc. of IEEE International Workshop on Information Assurance (IWIA)*, 2006.

[6] A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7), 1997.

[7] A. Cherepanov. Win32/industroyer – a new threat for industrial control systems. Technical report, ESET, 2017.

[8] P. M. Comparetti, G. Wondracek, C. Kruegel, and E. Kirda. Prospex: Protocol specification extraction. In *Proc. of IEEE Symposium on Security and Privacy*, 2009.

[9] G. Cormode and S. Muthukrishnan. Approximating data with the count-min sketch. *Journal of IEEE Software*, 29(1), 2012.

[10] W. Cui, V. Paxson, N. C. Weaver, and R. H. Katz. Protocol-independent adaptive replay of application dialog. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2006.

[11] S. Dharmapurikar and V. Paxson. Robus TCP reassembly in the presence of adversaries. In *Proc. of USENIX Security Symposium*, 2005.

[12] V. L. Do, L. Fillatre, and I. V. Nikiforov. A statistical method for detecting cyber/physical attacks on SCADA systems. In *Proc. of IEEE Conference on Control Applications (CCA)*, 2014.

[13] H. Dreger, M. Mai, A. Feldmann, V. Paxson, and R. Sommer. Dynamic application-layer protocol analysis for network intrusion detection. In *Proc. of USENIX Security Symposium*, 2006.

[14] P. Düssel, C. Gehl, P. Laskov, J. Bußer, C. Störmann, and J. Kästner. Cyber-critical infrastructure protection using real-time payload-based anomaly detection. In *Proc. of of Critical Information Infrastructures Security CRITIS*, 2009.

[15] N. Falliere, L. O. Murchu, and E. Chien. W32.stuxnet dossier. Symantec Corporation, 2011.

[16] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8), 2006.

[17] C. Feng, T. Li, and D. Chana. Multi-level anomaly detection in industrial control systems via package signatures and lstm networks. In *Proc. of Conference on Dependable Systems and Networks (DSN)*, 2017.

[18] P. Fogla and W. Lee. Evading network anomaly detection systems: Formal reasoning and practical techniques. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 2006.

[19] P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee. Polymorphic blending attacks. In *Proc. of USENIX Security Symposium*, 2006.

[20] I. N. Fovino, A. Carcano, T. D. L. Murel, A. Trombetta, and M. Masera. Modbus/dnp3 state-based intrusion detection system. In *Proc. of IEEE International Conference on Advanced Information Networking and Applications (AINA)*, 2010.

[21] H. Gascon, C. Wressnegger, F. Yamaguchi, D. Arp, and K. Rieck. Pulsar: Stateful black-box fuzzing of proprietary network protocols. In *Proc. of Int. Conference on Security and Privacy in Communication Networks (SECURECOMM)*, 2015.

[22] C. Gates and C. Taylor. Challenging the anomaly detection paradigm: A provocative discussion. In *Proc. of New Security Paradigms Workshop (NSPW)*, 2006.

[23] D. Hadžiosmanović, L. Simionato, D. Bolzoni, E. Zambon, and S. Etalle. N-gram against the machine: On the feasibility of the n-gram network analysis for binary protocols. In *Proc. of International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2012.

[24] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel. Through the eye of the PLC: semantic security monitoring for industrial processes. In *Proc. of Annual Computer Security Applications Conference (ACSAC)*, 2014.

[25] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6, 1979.

[26] K. L. Ingham and H. Inoue. Comparing anomaly detection techniques for HTTP. In *Proc. of International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2007.

[27] I. Kiss, B. Genge, and P. Haller. A clustering-based approach to detect cyber attacks in process control systems. In *Proc. of IEEE International Conference on Industrial Informatics (INDIN)*, 2015.

[28] G. Koutsandria, V. Muthukumar, M. Parvania, S. Peisert, C. McParland, and A. Scaglione. A hybrid network IDS for protective digital relays in the power transmission grid. In *Proc. of IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2014.

[29] M. Krotofil, J. Larsen, and D. Gollmann. The process matters: Ensuring data veracity in cyber-physical systems. In *Proc. of ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, 2015.

[30] C. Kruegel, T. Toth, and E. Kirda. Service specific anomaly detection for network intrusion detection. In *Proc. of ACM Symposium on Applied Computing (SAC)*, 2002.

[31] T. Krueger, N. Kraemer, and K. Rieck. ASAP: Automatic semantics-aware analysis of network payloads. In *Proc. of ECML Workshop on Privacy and Security Issues in Machine Learning*, 2010.

[32] T. Krueger, H. Gascon, N. Kraemer, and K. Rieck. Learning stateful models for network honeypots. In *Proc. of ACM Workshop on Artificial Intelligence and Security (AISEC)*, 2012.

[33] K. Lab. The DUQU 2.0 – technical details. Technical report, Kaspersky Lab, 2015.

[34] C. Leita, K. Mermoud, and M. Dacier. ScriptGen: An automated script generation tool for honeyd. In *Proc. of Annual Computer Security Applications Conference (ACSAC)*, 2005.

[35] Z. Lin, X. Jiang, and D. Xu. Automatic protocol format reverse engineering through context-aware monitored execution. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2008.

[36] M. Luchs and C. Doerr. Last line of defense: A novel ids approach against advanced threats in industrial control systems. In *Proc. of Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2017.

[37] F. Miao, Q. Zhu, M. Pajic, and G. J. Pappas. Coding sensor outputs for injection attacks detection. In *Proc. of IEEE Conference on Decision and Control (CDC)*, 2014.

[38] Y. Mo, R. Chabukswar, and B. Sinopoli. Detecting integrity attacks on SCADA systems. *IEEE Transactions on Control Systems Technology (TCST)*, 22(4), 2014.

[39] Modbus.org. Modbus application protocol specification v1.1b3. Technical report, Modbus.org, 2012.

[40] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48, 1970.

[41] R. Pang, V. Paxson, R. Sommer, and L. L. Peterson. binpac: a yacc for writing application protocol parsers. In *Proc. of Internet Measurement Conference (IMC)*, 2006.

[42] M. Parvania, G. Koutsandria, V. Muthukumar, S. Peisert, C. McParland, and A. Scaglione. Hybrid control network intrusion detection systems for automated power distribution systems. In *Proc. of Conference on Dependable Systems and Networks (DSN)*, 2014.

[43] F. Pasqualetti, F. Dörfler, and F. Bullo. Attack detection and identification in cyber-physical systems. *IEEE Transactions on Automatic Control*, 58 (11), 2013.

[44] R. Perdisci, D. Dagon, W. Lee, P. Fogla, and M. I. Sharif. Misleading worm signature generators using deliberate noise injection. In *Proc. of IEEE Symposium on Security and Privacy*, 2006.

[45] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee. McPAD: A multiple classifier system for accurate payload-based anomaly detection. *Computer Networks*, 5(6), 2009.

[46] M. Polychronakis, K. G. Anagnostakis, and E. P. Markatos. Comprehensive shellcode detection using runtime heuristics. In *Proc. of Annual Computer Security Applications Conference (ACSAC)*, 2010.

[47] S. S. Response. W32.duqu – the precursor to the next stuxnet. Technical report, Symantec, 2011.

[48] K. Rieck and P. Laskov. Detecting unknown network attacks using language models. In *Proc. of Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2006.

[49] K. Rieck and C. Wressnegger. Harry: A tool for measuring string similarity. *Journal of Machine Learning Research (JMLR)*, 17(9), 2016.

[50] M. Rocchetto and N. O. Tippenhauer. Towards formal security analysis of industrial control systems. In *Proc. of ACM Asia Conference on Computer and Communications Security (ASIA CCS)*, 2017.

[51] S. Rubin, S. Jha, and B. P. Miller. Automatic generation and analysis of NIDS attacks. In *Proc. of Annual Computer Security Applications Conference (ACSAC)*, 2004.

[52] F. Schuster, A. Paul, R. Rietz, and H. König. Potentials of using one-class SVM for detecting protocol-specific anomalies in industrial networks. In *Proc. of IEEE Symposium Series on Computational Intelligence (SSCI)*, 2015.

[53] K. Z. Snow, S. Krishnan, F. Monrose, and N. Provos. ShellOS: Enabling fast detection and forensic analysis of code injection attacks. In *Proc. of USENIX Security Symposium*, 2011.

[54] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *Proc. of IEEE Symposium on Security and Privacy*, 2010.

[55] Y. Song, M. E. Locasto, A. Stavrou, and S. J. Stolfo. On the infeasibility of modeling polymorphic shellcode. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 2007.

[56] Y. Song, A. Keromytis, and S. Stolfo. Spectrogram: A Mixture-of-Markov-Chains Model for Anomaly Detection in Web Traffic. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2009.

[57] Y. Song, M. E. Locasto, A. Stavrou, A. D. Keromytis, and S. J. Stolfo. On the infeasibility of modeling polymorphic shellcode: Re-thinking the role of learning in intrusion detection systems. *Machine Learning*, 81 (2), 2010.

[58] A. Teixeira, S. Amin, H. Sandberg, K. H. Johansson, and S. S. Sastry. Cyber security analysis of state estimators in electric power systems. In *Proc. of IEEE Conference on Decision and Control (CDC)*, 2010.

[59] The Bro Project. Bro – network security monitor. https://www.bro.org/index.html, 2017.

[60] R. Udd, M. Asplund, S. Nadjm-Tehrani, M. Kazemtabrizi, and M. Ekstedt. Exploiting bro for intrusion detection in a SCADA system. In *Proc. of ACM International Workshop on Cyber-Physical System Security*, 2016.

[61] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg. Limiting the impact of stealthy attacks on industrial control systems. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 2016.

[62] G. Vigna, W. Robertson, and D. Balzarotti. Testing network-based intrusion detection signatures using mutant exploits. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 2004.

[63] N. Šrndić and P. Laskov. Detection of malicious PDF files based on hierarchical document structure. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2013.

[64] O. Vukovic and G. Dán. On the security of distributed power system state estimation under targeted attacks. In *Proc. of ACM Symposium on Applied Computing (SAC)*, 2013.

[65] K. Wang and S. J. Stolfo. One-class training for masquerade detection. In *Proc. of ICDM Workshop on Data Mining for Computer Security*, 2003.

[66] K. Wang and S. J. Stolfo. Anomalous payload-based network intrusion detection. In *Proc. of International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2004.

[67] K. Wang, G. Cretu, and S. J. Stolfo. Anomalous payload-based worm detection and signature generation. In *Proc. of International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2005.

[68] K. Wang, J. J. Parekh, and S. J. Stolfo. Anagram: A content anomaly detector resistant to mimicry attack. In *Proc. of International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2006.

[69] Y. Wang, Z. Xu, J. Zhang, L. Xu, H. Wang, and G. Gu. SRID: state relation based intrusion detection for false data injection attacks in SCADA. In *Proc. of European Symposium on Research in Computer Security (ESORICS)*, 2014.

[70] G. Wondracek, P. M. Comparetti, C. Kruegel, and E. Kirda. Automatic network protocol analysis. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2008.

[71] C. Wressnegger, G. Schwenk, D. Arp, and K. Rieck. A close look on n-grams in intrusion detection: Anomaly detection vs. classification. In *Proc. of ACM Workshop on Artificial Intelligence and Security (AISEC)*, 2013.

[72] C. Wressnegger, F. Yamaguchi, D. Arp, and K. Rieck. Comprehensive analysis and detection of flash-based malware. In *Proc. of Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2016.

[73] D. Yang, A. Usynin, and J. W. Hines. Anomaly-based intrusion detection for scada systems. In *Proc. of intl. topical meeting on nuclear plant instrumentation, control and human machine interface technologies*, 2006.