Reading Between the Fields: Practical, Effective Intrusion Detection for Industrial Control Systems

Ömer Yüksel[1]
o.yuksel@tue.nl

Jerry den Hartog[1]
j.d.hartog@tue.nl

Sandro Etalle[1 2 3]
s.etalle@tue.nl

[1]Eindhoven University of Technology, The Netherlands
[2]University of Twente, The Netherlands
[3]SecurityMatters BV, The Netherlands

ABSTRACT

Detection of previously unknown attacks and malicious messages is a challenging problem faced by modern network intrusion detection systems. Anomaly-based solutions, despite being able to detect unknown attacks, have not been used often in practice due to their high false positive rate, and because they provide little actionable information to the security officer in case of an alert. In this paper we focus on intrusion detection in industrial control systems networks and we propose an innovative, practical and semantics-aware framework for anomaly detection. The network communication model and alerts generated by our framework are user-understandable, making them much easier to manage. At the same time the framework exhibits an excellent trade-off between detection rate and false positive rate, which we show by comparing it with two existing payload-based anomaly detection methods on several ICS datasets.

CCS Concepts

•**Security and privacy** → **Intrusion detection systems;**

Keywords

Industrial control systems, Anomaly detection

1. INTRODUCTION

Industrial control systems (ICS) have been used to control processes in a variety of areas such as water and energy distribution, manufacturing and transportation. Protection of ICS networks has become an increasingly important topic of research triggered by several security incidents over the last decade and the potential damage involved. Preventative approaches based on access control may be undesirable in scenarios where the availability of the system is critical.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

SAC 2016,April 04-08, 2016, Pisa, Italy

©2016 ACM. ISBN 978-1-4503-3739-7/16/04…$15.00

DOI: http://dx.doi.org/10.1145/2851613.2851799

Intrusion detection systems (IDS), which raise alerts about suspicious activities for a human operator to investigate, are required in such cases.

IDSs can be divided into two main categories by their means of detection: misuse and anomaly based [27]. Misuse based approaches attempt to recognize known attacks. Anomaly based systems, on the other hand, create a model of the system's regular behavior to detect abnormal activities. Misuse-based systems have a relatively low false positive rate and have, therefore, so far been prevalent amongst deployed systems. However, when it comes to protecting the critical infrastructure, one is particularly afraid of two things, neither of which can be detected by misuse-based systems: first, attacks exploiting previously unknown ('zero day') vulnerabilities [19]; secondly, attacks such as *process-related* ones – often carried out by insiders – targeting the activities of the ICS by malicious use of legitimate commands sent to field devices. Anomaly-based methods are required to detect such attacks.

In network traffic, detection of both types of attacks require inspecting the actual communication content (the 'payload'), as they may consist of a single malicious message and can escape detection by flow-based approaches. Stuxnet malware, for example, utilized protocols such as SMB, MS-RPC, HTTP, MS-SQL and S7 during its execution [11], which are all encapsulated as the payload of TCP segments. However, existing anomaly-based approaches proposed for the payload, which regard the content as a byte string, are shown to create too many false positives for practical use on ICS network traffic over binary protocols [14]. Moreover, since these methods are based on hard-to relate to statistical models, they result in a 'black-box' model: it is difficult for a human operator to understand and adjust the model, and the alerts created by the IDS typically do not give an insight on the underlying cause.

Our contribution is a new, practical type of anomaly-based IDS for industrial control systems that proves to be effective, generating few false alerts and higher detection rates than competing approaches on various reconnaissance, system- and process-related attacks. We demonstrate the framework's effectiveness by evaluating its performance on several ICS datasets.

The framework is based on a user-understandable ('white-box') model based on probability mass functions, allowing

an operator to tune the system to further reduce false alerts and to understand the nature of alerts when they occur; this is needed to react timely to an attack and to distinguish real attacks from the few remaining false alerts. The main innovation making the system effective is its rich, semantics-aware feature set: we use semantic information about the protocol and the domain, such as the type and interpretation of a field and expected or observed interdependencies between fields, to construct features that capture the meaning of messages rather than only their format. We apply our approach to Modbus RTU [15], Modbus-TCP [26], and Siemens S7 [25], that is three of the mostly widely adopted protocols in the industrial control systems, however, we are confident that the method can be applied to industrial control systems using other protocols as well. In addition, the framework is scalable to large networks and datasets and can be updated on run-time, making it viable for use in practice.

The remainder of the paper is organized as follows. In Section 2 we introduce the general framework for intrusion detection and in Section 3 we provide its instantiation on ICS networks. In Section 4 we present the details of the evaluation of the framework, including the datasets, payload-based approaches chosen for comparison and detection results. In Section 5 we present the framework's computational performance, followed by discussion of our findings in Section 6. We describe the related work in Section 7 and briefly conclude in Section 8.

2. GENERAL FRAMEWORK

In this section we illustrate the general framework behind the practical system that will be introduced in the next section. We will show that this general framework (i) is simple and easy for a user to understand and modify, yet (ii) supports different types of features making it highly customizable and thus applicable to many areas, and, most importantly, (iii) is effective at detecting malicious traffic while maintaining low false positive rates.

The starting point for our model is *messages* within the system. The nature of a message is problem domain dependent. In a network, for example, it could be an application layer message. Here we abstract away from such details and simply define messages as objects without further interpretation. In the sequel we refer to a fixed but unspecified set of possible messages M_S and use m to range over M_S. The network traffic consists of normal messages, which make up the majority of the traffic, and malicious messages.

Definition 1. Normal traffic. Normal traffic is a random element N defined over M_S, with probability mass function $P(N)$ denoting the distribution of normal messages in M_S.

Directly estimating the distribution of normal messages to distinguish them from malicious messages is infeasible. Therefore, we consider the *features* of a message that capture its relevant properties. Examples of features are the destination IP address of a message, the function being called on that destination, an argument to that function, etc. In this paper we will consider *numerical* features giving e.g. integers or floats and *nominal* features yielding for example names. In addition to these *elementary features*, we also consider *compound features* which give tuples of numeric and/or nominal values. Together all features form the so called *feature vector*, which extracts *value vectors* out of the messages.

Definition 2. Feature and value vector. Let m be a message in M_S.

- A *feature* f is a function $f : M_S \to V_f$ mapping messages into a *feature domain* V_f
- Given a sequence of features $D = (f_1, ..., f_k)$ with domains $(V_1, \ldots, V_k)$, the *compound feature* f^D has domain $V_1 \times \ldots \times V_k$ and is defined by $f^D(m) = (f_1(m), ..., f_k(m))$.

A *feature vector* is a tuple of features $\overline{f} = (f_1, ..., f_n)$. Given $\overline{f}$, we call $\overline{f}(m) = (f_1(m), \ldots, f_n(m))$ the *value vector* of m.

For example, a feature vector ('connection', 'function_code', 'parameter_1') would yield the value vector ((10.0.1.10:1500, 10.0.1.20:80), 5, 15) for the message that calls function 5 with argument 15 over the given connection.

Malicious messages may have the same values as normal messages on some features. However, if we choose the right features, they will have values that are rare amongst normal traffic on at least one of the features. For features that can take many different values, such as numeric features, individual values may be rare even for normal traffic. Yet rare values should indicate that a message is likely an attack. Therefore we group values into so called *bins* and consider the occurrence of bins rather than individual values.

Definition 3. Binning vector. A *binning* $B = \{b_1, b_2, \ldots\}$ for a feature f is a partitioning of the feature domain V_f. The bin $B(v)$ of a value v is the bin in which that value falls; $B(v) = b$ when $v \in b, b \in B$. A *binning vector* $\overline{B} = (B_1, \ldots, B_n)$ for a feature vector $(f_1, \ldots, f_n)$ comprises a binning B_i for each feature f_i $(i = 1, \ldots, n)$.

For nominal features we usually consider the binning where each bin consists of a single element, for numeric features we usually consider bins that are ranges $[v_l, v_u)$ and for compound features we consider bins that are the product of such bins. We address details of bin selection in the next section on instantiation of our framework.

Our fundamental assumption states that for attack messages a feature will yield bins with low probability. We are therefore looking for *anomalies*, messages where the probability of some feature is below a given threshold.

Definition 4. Anomalous bins and values. Let $P(N)$ be a distribution of normal traffic, and B a binning for the feature f. We define the *probability of a bin* $b \in B$ as the probability that normal traffic is mapped by f into b and we consider the bin, and all values in it, anomalous if this probability is below a given *threshold* t:

- $prob_N(b) = P(f(N) \in b)$
- $(v \in) b$ is anomalous if $prob_N(b) < t$

Our main goal is to find anomalies along with their causes so they can be presented to an operator whom can then evaluate and address them.

We could define anomalous messages as those yielding an anomalous value on one of the features. However, not all entities on a system behave the same. Thus what is normal for one entity may be anomalous for another. To address this we introduce *profiles* which are conditional distributions describing the normal traffic per entity. We use a so called *profiling feature*, which we assume is the first feature in a feature vector, to determine the 'entity' for a message. In this way we can for example make a profile for each source IP or for each connection. A message is anomalous when the entity involved is anomalous or when one of the features of the message is anomalous for that entity.

Definition 5. Profile & Anomalous messages. Let $(f_1, \ldots, f_n)$ be a feature vector with binning vector $(B_1, \ldots B_n)$ and N the random element representing normal traffic. The *profile* for $b \in B_1$ (and all values $v \in b$) is random element $N|b$ with distribution $P(N|f_1(N) = b)$.

A message m with value vector $\overline{f}(m) = (v_1, \ldots, v_n)$, is anomalous for a given threshold t if v_1 is anomalous or any of the other values are anomalous within the profile for v_1, i.e. m is *anomalous* if:

$$\exists b_1 \in B_1 : v_1 \in B_1 \wedge (\mathit{prob}_N(b_1) < t \vee$$
$$\exists j \in \{2, \ldots, n\}, b_j \in B_j : v_j \in b_j \wedge \mathit{prob}_{N|b_1}(b_j) < t)$$

If v_j is anomalous we call the corresponding feature f_j a *cause* of the anomaly.

Here we have used a single global threshold t for all features and bins, however one can also use a threshold per feature or even per individual bin. Note that $\mathit{prob}_N(b_1) < t$ indicates an anomaly on the profiling feature, for example, a message with an unusual source IP is observed in the traffic while $\mathit{prob}_{N|b_1}(b_j) < t$ indicates an anomaly within a profile, such as a function not usually called from this source IP. The choice of a profiling feature is based on its ability to distinguish between sets of behavior. In industrial control systems, for example, devices tend to follow different behavior patterns, which would make device identifiers suitable profiling features.

This completes the description of the general framework and the profiles. We provide an instantiation for industrial control systems traffic in the next section.

3. AN EFFECTIVE MONITORING SYSTEM

We can finally introduce the intrusion detection system we are aiming to by applying the general framework illustrated in the previous section. In this section we are going to explain the features we have to select for a monitoring that is practical, effective, and has a (very) low number of false positives, as it will be shown in the next section. Recall that we aim at detecting both system- as well as process-based threats.

In the sequel, we first describe the setting and the threat model, then the protocols involved in our instantiation, and finally the details of selecting features, corresponding bins and thresholds. In the remainder of this paper we also refer to our instantiation as the white-box framework.

3.1 Setting and Threat Model

ICSs have several components connected over computer networks, for example programmable logic controllers (PLCs) which are specialized computers used to automate the operations, human-machine interfaces (HMIs) which allow operators to interact with other components of the system, various workstations, and control servers [16].

In this setting, the assets we wish to protect are the network *hosts* such as PLCs and HMIs and *field devices* in the system (controlled by the hosts) such as valves, pumps and heaters. The industrial processes, e.g. transportation of gas through a pipeline, are controlled through these hosts and devices.

Potential threats against the assets are classified as *system-related* and *process-related* in [13]. In addition, we consider *reconnaissance* against the hosts. System-related threats are typically low-level and are associated with the vulnerabilities of the network hosts, such as stopping the processor on a PLC. Process-related threats are associated with field devices, and include the attack messages that are legitimate on the application level, but disruptive to ICS activities, such as raising the temperature to a critical point in a heater. Reconnaissance includes extracting device configuration and scanning the PLCs for memory addresses or function codes. We focus on malicious messages on ICS protocols and do not consider threats outside this scope, e.g. attacks on lower layers such as TCP.

3.2 Protocols

With our threat model, detecting the majority of the attacks requires inspection of the ICS-specific protocol fields. As such, an understanding of the protocols is necessary to create the right features. For our experiments, we consider Modbus-RTU over a serial connection, Modbus-TCP, and S7, a protocol for Siemens devices.

Modbus is an open standard ICS-specific protocol by Schneider Electric [26]. Modbus-TCP and Modbus-RTU are two common variants of the protocol, the former can only be deployed over TCP/IP stack, while the latter also runs over serial connection. Communication is based on a simple client-server model, typically with a single master (client) device and multiple slave (server) devices.

S7 is a proprietary protocol by Siemens mainly used for communication of PLCs in ICS networks [25]. Compared to Modbus, S7 is more complex, with a larger number of fields. Official protocol specification is not public, however, considerable amount of information about S7 is revealed through reverse engineering and a publicly available Wireshark plug-in can be used to parse S7 messages [4].

3.3 Feature Selection

Selecting the correct set of features is essential for the accuracy of any IDS. Here we derive elementary and compound features for the messages in the protocols above. We obtain elementary features from message fields interpreted according to protocol semantics and (if available) domain knowledge. We mainly extract features from ICS protocol fields, which contain information such as the type of command, location or address the host is accessing, the values that are

ICS Protocol	Encapsulating protocols	Excluded fields
Modbus-RTU	-	CRC
Modbus-TCP	IP	trans_id
S7	IP, COTP, Ethernet	cotp.segment, header.pduref, srcref resp.data, param.userdata.seq_num

Table 1: Fields excluded at feature selection phase.

being changed or returned. For instance, the field 'reference number'(register address) in Modbus forms an elementary feature and will generate alerts for use of an unusual register which could indicate reconnaissance (reading an unusual register), or an attack against the system (writing to an unusual register). Fields from low-level protocols are also selected if they yield useful features such as device identification and message length.

We select most but not all ICS protocol fields in building features. We need features that yield rare bins in case of attacks, but not for normal traffic. We therefore do not select fields that are incremental such as transaction ID or behave similar to random numbers such as nonces, numeric fields with high variance and nominal fields with high entropy.

Overall we select 18, 25 and 216 elementary features for Modbus-TCP, Modbus-RTU and S7 respectively. Due to space limitations we refer to [5] for features extracted for Modbus-RTU (see Section 4.1 on datasets), and the appropriate Wireshark fields [4, 6] for Modbus-TCP and S7. Table 1 contains the list of excluded fields.

Compound features are constructed from sequences of elementary features with dependencies. A general discussion of selection methodology for compound features is beyond the scope of this work. Instead we only demonstrate their usefulness in our tests on Modbus-RTU traffic. We consider pairs of elementary features that have a semantic relation and confirm their relationship within the normal traffic using Pearson's Chi-Square test [22] for statistical dependencies. In particular, we form all pairs of elementary features with 'delta' counterparts, such as (SetPoint, deltaSetPoint), in which the latter field indicates the change in the value.

In summary, we derive our features by syntactic and semantic interpretation of the messages. The expert selects the features according to protocol and domain knowledge, and an exploration of the normal traffic.

3.4 Bin and Threshold Selection

Default bin size of a numeric feature is automatically computed by taking a sample of normal traffic, cleaning the sample by discarding outliers, and applying Scott's rule [24]. The threshold provides a trade-off between false alerts and detection rate, with a higher value also providing more robustness against noise (attacks) in the training set. By default, a single threshold t is applied to all elementary features, and t^n to compound features constructed from n dependent features. Our framework also allows a human expert to manually select the bins, or apply specific thresholds to selected features or bins.

3.5 White-box Model and Tuning

An advantage of a simple model with meaningful features

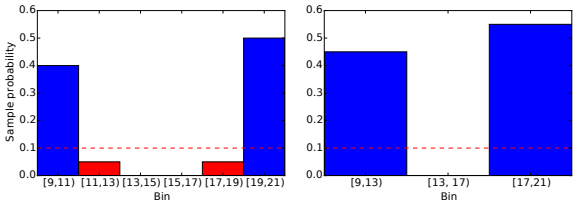

Figure 1: Distribution of bins in the normal traffic with different binning schemes, with threshold $t = 0.1$.

is that it can be viewed and adjusted (*tuned*) by a human according to his/her knowledge of the system. For example, from the model for the feature 'payload length' (Modbus-TCP), shown in Figure 1, the expert sees that the values in the bins $[11, 13)$, $[13, 15)$, $[15, 17)$ and $[17, 19)$ are considered anomalous. The expert realizes that the system is too strict because these values, while not common, are normal. Choosing wider bins, as shown in the right-hand side of the figure, or lowering the threshold to $t = 0.05$, are two options to address this. In both cases, alerts will be created for values in the range $[13, 17)$ but not for $[11, 13)$ or $[17, 19)$.

In the next section we describe the experiments we use to evaluate our white-box framework for ICS.

4. PRACTICAL EVALUATION

We evaluate our framework on datasets with Modbus-RTU, Modbus-TCP and S7 traffic. For each we use the features selected in the previous section to create a model of the normal traffic as described in Section 2. We test each model with both normal traffic and attack traffic to evaluate the false positive rate and detection rate respectively.

We compare the results to two previous payload-based approaches; attributed token kernel over one-class support vector machines (SVM-ATK) [10] and McPad [21]. Both approaches utilize n-gram analysis, i.e. they model the whole or parts of the payload as byte-sequences with the length n. In this section we describe the datasets, the approaches and their configuration, and the evaluation methods and the results of the evaluation.

4.1 Datasets

In this section we describe the three datasets used to evaluate the detection performance of the methods; a public dataset with Modbus-RTU traffic, and two private datasets, one with Modbus-TCP and one with S7 protocol traffic.

Modbus-RTU. This publicly available dataset is provided by the Critical Infrastructure Protection Center of the Mississippi State University [5]. It captures communication over a serial connection using the Modbus-RTU protocol, between an RTU and a PLC in a gas pipeline. The dataset contains normal communication as well as reconnaissance and process-related attacks.

We categorize the attacks in the dataset into four subsets. *Scanning* denotes the address and function code scan attacks. *Illegal value* denotes negative values, illegal setpoint, single data injection and set value injection. *Timing* attacks include slow change, burst, fast change and value wave injec-

Dataset	Type	Subcategory	#Messages
Modbus-RTU	Normal		56156
	Attack	Timing	916
		Illegal data	521
		Illegal command	49
		Scan	11
Modbus-TCP	Normal		100000
	Attack	Device info	4
		Reconfiguration	8
		Scan	141
		Commands	6
S7	Normal		50000
	Attack	Scan	156
		CPU cycle	5148

Table 2: Summary of the datasets used for evaluation.

tion. Lastly, *illegal command* denotes the illegal PID command attack.

Modbus-TCP. The normal dataset contains traffic samples provided by a company, captured from a water tank simulator hosted within a virtual machine. The network has a single master and single slave device communicating using the Modbus-TCP protocol.

We use publicly available reconnaissance and system-related attack samples from an ICS security company [1] to evaluate the detection performance. We classify the attack messages into four categories: *unauthorized commands*, which include read and write requests to unexpected registers within the system, *device reconfiguration*, *scanning* for addresses and function codes, and attempts to extract *device information*.

S7. The normal traffic was sampled from a network belonging to a real-life industrial control system in a governmental organization, with multiple hosts communicating over S7 protocol for Siemens devices. Due to the confidentiality agreements, we are not able to disclose any further details regarding the system itself.

We use publicly available attack datasets for evaluation [3] with two categories: *PLC scanning* for reconnaissance and *CPU cycle*, a system-related attack exploiting a vulnerability that allows issuing administrative commands [2].

4.2 Validation

We validate our framework by evaluating its performance against previous anomaly-based works in literature. Here we focus on the closest competing approaches: payload-based approaches that can be applied to general network traffic. Thus we consider SVM-ATK by Düssel [10] et al., which utilizes a protocol parser, and McPad [21], which is a purely n-gram based method also using SVMs. Below we describe these approaches and their configuration, as well as the details of profiling and tuning in our experiments.

Profiling and Tuning. For our framework, explained in previous sections, we test all datasets with default binning. Additionally we demonstrate the advantages of tuning the binning, possible due to the white-box nature, on the Modbus-RTU dataset (this procedure is explained in Section 3.5). This manual tuning takes a numeric feature with bins below the threshold $t = 0.01$ and modifies the bin range so only 'obvious outliers' fall in anomalous bins.

The framework also requires selecting a profiling feature. We use a (natural) profiling feature that distinguishes between different devices: the Command/Response value for Modbus-RTU, source IP for Modbus-TCP and source MAC address for S7 dataset.

SVM with attributed token kernel. This method, proposed by Düssel et al. [10], builds its features from tokens extracted from application-layer messages using a parser. The 'attributed token kernel' (ATK) function returns the weighted sum of n-gram similarity between tokens that are shared by two application-layer messages. This function is used as the kernel of a one-class support vector machine (SVM), which models the normal data by establishing its boundaries in a vector space. The approach requires quadratic time and space as it needs kernel matrix for training and detection, which contains pairwise similarity values for all messages in the training data.

The approach provides a higher accuracy than the regular SVM kernels for n-gram comparison. While the approach is evaluated with HTTP datasets, it is protocol independent, and the authors proposed to apply the method on other domains such as ICS traffic. To the best of our knowledge, it is the only general-purpose approach in literature that utilizes an application layer protocol parser, which motivated our choice of comparison. We implemented the kernel as described in [10]. As in the original work, we use the n-gram length of 1. The approach does not include a supervised 'feature selection' phase, however, we use the same set of fields as white-box framework for a fair comparison.

McPad. Perdisci et al. [21] propose McPad (Multiple Classifier System for Accurate Payload-based Anomaly Detection), an n-gram based method that also uses one-class SVMs. McPad addresses the problem of the feature space increasing exponentially with the value of n, the substring length. Instead of higher-order n-grams, the authors propose using pairs of bytes that have the distance of v, denoted 2_v-grams. Separate feature vectors are constructed by varying the parameter v, which are then used to construct multiple SVM-based classifiers. During detection, each classifier marks a payload as normal or anonymous, and a combination rule determines whether an alert is generated.

We use the implementation of McPad made publicly available by the authors. For the selectable parameters we stick to the defaults which give the best result in the original work: 160 as the number of clusters and 'maximum probability' as the combination rule. Since the method requires the raw payload to run reliably, we do not test McPad on Modbus-RTU dataset, which is in plain-text.

4.3 Evaluation Methodology

The general method to evaluate an approach is to split the normal dataset into two disjoint subsets, use one to build a model of the normal traffic (*training set*), and the other subset to test for false alerts (denoted *normal test set*), then perform detection on attack data. Thus the performance metrics we use are *detection rate*(DR), the ratio of detected attacks to the size of attack dataset, and *false positive rate* (FPR), the ratio of alerts raised on normal messages to the size of the normal test set. All of the methods we evaluate

contain parameters to control the trade-off between FPR and DR. We repeat the experiments with varied parameters to obtain the efficiency of the trade-off for each method.

We evaluate McPad and our framework using *n-fold cross validation* method: the normal dataset is randomly partitioned into n distinct subgroups, called folds. For each fold, training is performed on the union the remaining folds, test for FPR is performed on the fold itself, and test for DR is performed on the entire attack dataset. The mean values of FPR and DR over n tests yield the overall detection performance. We opt for a value of $n = 5$, which produces stable results of DR and FPR in preliminary tests.

As previously mentioned, SVM-ATK has limitations on dataset size. Therefore we evaluate it by repeated subsampling (as performed by its authors) instead of n-fold cross-validation. First, two disjoint subsets of size n are randomly sampled from the normal dataset, to be used for training and testing. The model is created using the training set, and evaluated on the test set and the complete attack dataset. The process is repeated k times, and the mean DR and FPR are selected as the performance metrics. The authors used a training sample size $n = 500$ and the number of repetitions $k = 50$. We opt for $n = 1000$ to provide a more accurate detection at the cost of computational performance, and keep the same number of repetitions.

Finally, we repeat the experiments for McPad and SVM-ATK on filtered (response- or request-only) datasets in order to observe the difference made by such a 'semantic aid'.

4.4 Results

In this section we first present the results of our white-box framework with elementary features along with comparison to other methods, then the improvement provided by the compound features before moving on to discussion of the results in the next section.

Comparison of the Methods. We present the results in Table 3. FPR and the overall DR are considered as the main performance metrics. We also include the detection rate per attack type for completeness. As a rule-of-thumb, we select results with the configurations yielding overall $DR > 90\%$ with lowest FPR, but we also include the other results if there is a noticeable 'jump' in the trade-off.

In all three datasets, the white-box framework significantly outperforms the other methods in terms of FPR-DR trade-off. Between the SVM-based approaches, SVM-ATK outperforms McPad in S7 and Modbus-TCP protocols with the complete datasets. Filtering provides an improvement in the trade-off for both McPad and SVM-ATK. The change is more noticeable on McPad, with better performance than SVM-ATK on both datasets and yielding a 93.7% detection with no false positives on Modbus-TCP requests.

The white-box framework displays the anomalous features along with the alerts created. With this information the user can tune the model, i.e. change the bins or thresholds, to reduce false positives. For Modbus-TCP feature register values ('register_uint16') with default bins leads to false positives. Inspecting these features shows that the created bins are too narrow. Manually adjusting the bins of this feature

Dataset/Method	FPR	DR				
		Overall	Subcategory			
Modbus-RTU			Scan	Timing	Data	Cmd.
White-box	0.08%	97.3%	100%	95.6%	100%	100%
	16.7%	100%	100%	100%	100%	100%
SVM-ATK (all)	27%	91%	92%	94%	84%	100%
SVM-ATK (requests)	12%	99.6%	100%	-	99.5%	100%
SVM-ATK (responses)	10.8%	70%	-	64%	86%	-
	60%	98%	-	97%	99%	-
Modbus-TCP			Scan	Device	Reconf.	Cmd.
White-box (default bins)	0.02%	100%	100%	100%	100%	100%
White-box (user bins)	0%	100%	100%	100%	100%	100%
SVM-ATK	38%	99.4%	100%	100%	100%	83.3%
SVM-ATK (requests)	5%	89.8%	92%	50%	100%	50%
	40%	100%	100%	100%	100%	100%
McPad (requests)	0%	93.7%	94.3%	50%	100%	100%
McPad (all)	91.3%	98.1%	99.3%	50%	100%	100%
S7			Scan	Cycle		
White-box	0.04%	100%	100%	100%		
SVM-ATK (all)	27%	68.8%	49.4%	69.4%		
	43%	89.7%	81%	90%		
SVM-ATK (requests)	33%	99.9%	99.8%	99.9%		
McPad	61%	100%	100%	100%		
McPad(requests)	20.2%	100%	100%	100%		

Table 3: Summary of the detection results.

eliminates all false positives while maintaining a 100% DR on Modbus-TCP dataset.

While outperforming other approaches the white-box method cannot achieve a 100% DR for Modbus-RTU without a significant increase in the FPR. This signals the possible presence of multivariate anomalies. We thus investigate if compound features can improve its performance in this scenario.

Compound Feature Building. Compound features extend our framework's detection capabilities to interdependent features. Table 4 shows comparison of the results.

As compound features we consider pairs that are semantically related and whose interdependence is confirmed by the dataset (see Section 3.3). With these compound features we find attacks (from the *timing* category) that are otherwise missed at low FPR. Thus we obtain an improved result, 100% DR with a FPR of 0.57%. Our white-box approach directly tells us that it is feature (*PipelinePSI, deltaPipelinePSI*) that is able to detect them.

	FPR	DR
White-box (stand-alone)	0.08%	98.88%
	0.58%	99.07%
	16.7%	100%
White-box (with compound features)	0.10%	99.91%
	0.57%	100%

Table 4: Comparison of the results for Modbus-RTU dataset with compound feature building.

5. COMPUTATIONAL PERFORMANCE

The white-box framework performs training (i.e. counting bins in messages) in linear time to the dataset size and detection in constant time, thus has a lower computational complexity than many machine learning based approaches, including SVM-based methods with a non-linear kernel. This makes our framework, both for training and detection, more scalable to larger datasets as required in real-life settings.

Approach	t_{train}(ms)	t_{detect}(ms)
White-box	2,579	0.27
McPad	34,143	0.36
SVM-ATK	239,356	486

Table 5: Comparison of average training (1000 messages) and detection (single message) time of the approaches on a sample of S7 messages.

The latter is particularly important as it is performed on network traffic in near real-time.

In order to provide a better insight on the performance, we also compare the implementations of our framework and other approaches using a small training sample of 1000 messages. We use the S7 dataset because: (i) it is more complex than the other two protocols, thus providing the 'worst case' scenario for our framework, (ii) it belongs to a real world ICS. We implemented our framework and SVM-ATK on Python 2.7 and used Wireshark to parse network traffic data. We perform the comparison on a Linux-based virtual machine with a single 2.4 GHz reserved CPU and 16 GB memory. The results are shown in Table 5. Since we use preprocessed datasets, the results do not include the parsing time, which is 0.7 ms per message on average. Our framework performs training much faster than McPad and SVM-ATK. Detection time is slightly lower than McPad's when the parser overhead is ignored.

While the parsing time is the main bottleneck, the results, given the computational complexity, imply that our framework is as feasible as the existing payload-based methods for real-life use in terms of performance. Further improvements can be achieved by utilizing an optimized parser module. In addition, profiling allows our framework to be used in a distributed setting for larger networks by partitioning the data into manageable sizes. If a location identifier is used as the profiling feature, multiple instances of the framework can be deployed in different network segments while yielding the detection capabilities of a centralized framework.

6. DISCUSSION AND LIMITATIONS

The framework we propose exhibits a significantly better DR-FPR trade-off over the n-gram based approaches in all datasets. In addition, the model generated by the framework can be viewed and tuned by the end-user, and is able to display the features with anomalous values along with the alerts. The latter is useful information for the user that has to respond to the alert.

Considering numeric or nominal values rather than just byte strings on ICS protocols avoids two important issues with the n-gram based approaches: (i) an attack message that differs from a normal message only on a single key feature, e.g. function code, is often not considered anomalous by n-gram analysis while it should, (ii) a small differences in numerical values in normal traffic can be interpreted as a relatively large difference in the n-gram similarity score, resulting in false alerts. We suspect that the high false positive rates with McPad and SVM-ATK are caused by the problems indicated. In the case of our framework, (i) results in an alert on the attack message, (ii) is largely prevented by binning numeric values.

The semantics-aware approach we present, while providing improved results, also comes with certain limitations. The quality of the features in the framework highly depend on the parser. A parser created by reverse engineering, such as in the case of S7, is not as reliable as the one created for open protocols. In addition, attacks composed of multiple messages are not considered by the current model. Sequential models or building features from groups of messages can be considered to address this problem more thoroughly.

7. RELATED WORK

Both our approach and data leakage detection framework by Costante et al. [7] use a white-box model based on probability estimates, with technical differences in the binning, feature types, and profiling. However, the approaches differ in the threat model and the features. The database work uses a pre-determined set of features, whereas our framework uses those derived from a protocol parser and selected by the expert. The feature vector in our case varies according to the protocol, available parser, domain knowledge and exploration of the normal traffic data.

Several open source network monitoring tools, such as Wireshark [6] and Bro [20], provide an application protocol parser. The latter can be used for intrusion detection with manually specified rules. Düssel et al. [10] propose an SVM-based approach that uses a protocol parser. We include this method in our experiments, with details and differences from our framework given in Section 4.2.

Dreger et al. [8] leverage Bro to extract features from application layer messages for signature-based detection in Internet traffic. Kruegel and Vigna [17] propose an anomaly detection method for web traffic partially utilizing application-layer information from HTTP requests.

In the area of ICS, Hadziosmanovic et al. [12] utilize Bro to extract register values in Modbus messages, and model the normal behavior of the values as time series. Lin et al. [18] propose a framework where DNP3 protocol traffic is analyzed to identify critical commands in ICS networks based on the function code and predict their consequences by using a state estimation method. Both aforementioned approaches, while providing the possibility to detect complex process-based attacks, require extensive knowledge of the underlying system to get the right features or construct the model. Düssel et al. [9] propose a payload-based anomaly detection method for ICS traffic, however does not utilize the application protocol syntax or semantics. Schuster et al. [23] propose using flow data, partial information from the ICS protocol fields, and message ordering for anomaly detection and implement their approach for Profinet IO protocol.

8. CONCLUSION AND FUTURE WORK

We introduced a highly customizable and practical framework for intrusion detection in industrial control systems. We evaluated the white-box framework against two other payload-based approaches on three datasets with different protocols: Modbus-RTU, Modbus-TCP and S7. Compared

to these n-gram based methods, the white-box framework provides a significantly better FPR/DR trade-off. Our results demonstrate the strength of leveraging semantics-based features, which allow distinguishing malicious messages from the normal ones while retaining a simple, user-understandable model. In future work, we plan to apply our approach to other settings such as back office networks. As our experiments show the importance of selecting good features (i.e. those that yield rare bins for attack messages) for the white-box approach, we also plan to focus on methods to support the expert in this task.

9. ACKNOWLEDGMENTS

This work has been supported by NWO, Dutch Organization for Scientific Research, through SpySpot project (no. 628.001.004); and by the European Commission through FP7-SEC-607093-PREEMPTIVE project funded by the 7th Framework Program.

10. REFERENCES

[1] Digital bond's scada security portal. http://www.digitalbond.com/tools/quickdraw/, 2011.

[2] Siemens s7-1200 plc vulnerabilities. https://ics-cert.us-cert.gov/alerts/ICS-ALERT-11-161-01, 2011.

[3] Desrics: Datasets enabling security research for industrial control systems. http://desrics.com/, 2014.

[4] S7comm wireshark dissector plugin. http://s7commwireshark.sourceforge.net/, 2014.

[5] J. Beaver, R. Borges-Hink, and M. Buckner. An evaluation of machine learning methods to detect malicious scada communications. In *International Conference on Machine Learning and Applications*, volume 2, pages 54–59, Dec 2013.

[6] G. Combs et al. Wireshark, 2015.

[7] E. Costante, J. den Hartog, M. Petković, S. Etalle, and M. Pechenizkiy. Hunting the unknown. In *Data and Applications Security and Privacy XXVIII*, pages 243–259. Springer, 2014.

[8] H. Dreger, A. Feldmann, M. Mai, V. Paxson, and R. Sommer. Dynamic application-layer protocol analysis for network intrusion detection. In *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, USENIX-SS'06, Berkeley, CA, USA, 2006. USENIX Association.

[9] P. Düssel, C. Gehl, P. Laskov, J.-U. Bußer, C. Störmann, and J. Kästner. Cyber-critical infrastructure protection using real-time payload-based anomaly detection. In E. Rome and R. Bloomfield, editors, *Critical Information Infrastructures Security*, LNCS 6027, pages 85–97. Springer, 2010.

[10] P. Düssel, C. Gehl, P. Laskov, and K. Rieck. Incorporation of application layer protocol syntax into anomaly detection. In R. Sekar and A. Pujari, editors, *Information Systems Security*, LNCS 5352, pages 188–202. Springer, 2008.

[11] N. Falliere, L. Murchu, and E. Chien. W32. stuxnet dossier. Technical Report November, 2011.

[12] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel. Through the eye of the plc: Semantic security monitoring for industrial processes. In *Proceedings of the 30th Annual Computer Security Applications Conference*, ACSAC '14, pages 126–135, New York, NY, USA, 2014. ACM.

[13] D. Hadziosmanovic, D. Bolzoni, S. Etalle, and P. Hartel. Challenges and opportunities in securing industrial control systems. In *Complexity in Engineering, 2012*, pages 1–6. IEEE, 2012.

[14] D. Hadziosmanovic, 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 D. Balzarotti, S. Stolfo, and M. Cova, editors, *Research in Attacks, Intrusions, and Defenses*, LNCS 7462, pages 354–373. Springer, 2012.

[15] Honeywell. Modbus rtu serial communications user manual, 2013.

[16] E. Knapp and J. Langill. *Industrial network security: securing critical infrastructure networks for smart grid, scada, and other industrial control systems.* Access Online via Elsevier, 2011.

[17] C. Kruegel and G. Vigna. Anomaly detection of web-based attacks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pages 251–261. ACM, 2003.

[18] H. Lin, A. Slagell, Z. Kalbarczyk, P. W. Sauer, and R. K. Iyer. Semantic security analysis of scada networks to detect malicious control commands in power grids. In *Proceedings of the First ACM Workshop on Smart Energy Grid Security*, SEGS '13, pages 29–34, New York, NY, USA, 2013. ACM.

[19] B. Miller and D. Rowe. A survey SCADA of and critical infrastructure incidents. *Proceedings of the 1st Annual conference on Research in information technology - RIIT '12*, page 51, 2012.

[20] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23):2435–2463, 1999.

[21] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee. McPAD: A multiple classifier system for accurate payload-based anomaly detection. *Computer Networks*, (October 2008).

[22] R. L. Plackett. Karl pearson and the chi-squared test. *International Statistical Review / Revue Internationale de Statistique*, 51(1):pp. 59–72, 1983.

[23] F. Schuster, A. Paul, and H. König. Towards learning normality for anomaly detection in industrial control networks. In *Emerging Management Mechanisms for the Future Internet*, pages 61–72. Springer, 2013.

[24] D. W. Scott. Scott's rule. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):497–502, 2010.

[25] Siemens. Cpu-cpu communication with simatic controllers, 2013.

[26] A. Swales. Open modbus/tcp specification. *Schneider Electric*, 29, 1999.

[27] S. Wu and W. Banzhaf. The use of computational intelligence in intrusion detection systems: A review. *Applied Soft Computing*, (November), 2010.