This is a Preprint Version

Intrusion and Anomaly Detection for the Next-Generation of Industrial Automation and Control Systems

Luis Rosa[a,*], Tiago Cruz[a], Miguel Borges de Freitas[a], Pedro Quitério[a], João Henriques[a,b], Filipe Caldeira[a,c], Edmundo Monteiro[a] and Paulo Simões[a]

[a] University of Coimbra, CISUC, Department of Informatics Engineering, 3030-290 Coimbra, Portugal
[b] Informatics Department, Polytechnic of Viseu, Viseu, Portugal
[c] CISeD –Research Centre in Digital Services, Polytechnic of Viseu, Portugal

ARTICLE INFO

Keywords:
IACS
Industrial Control Systems
SCADA
Cybersecurity
Critical Infrastructure Protection
Network Anomaly Detection
Intrusion Detection
Event Processing

ABSTRACT

The next-generation of Industrial Automation and Control Systems (IACS) and Supervisory Control and Data Acquisition (SCADA) systems pose numerous challenges in terms of cybersecurity monitoring. We have been witnessing the convergence of OT/IT networks, combined with massively distributed metering and control scenarios such as smart grids. Larger and geographically widespread attack surfaces, and inherently more data to analyse, will become the norm.

Despite several advances in recent years, domain-specific security tools have been facing the challenges of trying to catch up with all the existing security flaws from the past, while also accounting for the specific needs of the next-generation of IACS. Moreover, the aggregation of multiple techniques and sources of information into a comprehensive approach has not been explored in depth. Such a holistic perspective is paramount since it enables a global and enhanced analysis enabled by the usage, combination and aggregation of the outputs from multiple sources and techniques.

This paper starts by providing a review of the more recent anomaly detection techniques for SCADA systems, focused on both theoretical machine learning approaches and complete frameworks. Afterwards, it proposes a complete framework for an Intrusion and Anomaly Detection System (IADS) composed of specific detection probes, an event processing layer and a core anomaly detection component, amongst others. Finally, the paper presents an evaluation of the framework within a large-scale hybrid testbed, and a comparison of different anomaly detection scenarios based on various machine learning techniques.

1. Introduction

The latest generation of Industrial Automation and Control Systems (IACS), combining Industrial IoT (IIoT) and Supervisory Control and Data Acquisition (SCADA) environments, poses several challenges. According to a survey of two hundred automation executives in 2015 [63], the adoption of IIoT, primarily driven by the optimization of operational efficiency and productivity, faces cybersecurity as the biggest challenge. Similarly, in another study [60] more than two hundred industrial companies refer cybersecurity as a high priority. The importance of security across several Industry 4.0 enablers, including but not limited to Big Data, Artificial Intelligence and Open-Source Software, is also referred in another survey [4].

The attack surface of IACS has grown significantly over the past years. Major incidents, from Stuxnet [55] to Industroyer [21], keep showing their vulnerabilities, including the lack of security of SCADA communication protocols as one of the most criticized issues. Whereas this scenario is now changing, with several SCADA protocols being redesigned, it remains a problem, as legacy protocols are still widely used.

Using an energy smart grid as a reference scenario, this paper describes an Intrusion and Anomaly Detection System (IADS) specifically designed to tackle the architectural and security challenges of the next-generation of IACS. Namely, this paper presents a comprehensive strategy for monitoring both industrial network traffic and managed physical processes, as well as a way of integrating several heterogeneous components into a unified detection framework capable of monitoring the cybersecurity state of a SCADA system in near real-time.

The contributions of this paper are threefold. First, we provide an extensive literature review on SCADA anomaly detection, focused on the last four years. Second, we describe a complete framework for performing Intrusion and anomaly detection on IACS environments. Third, we present an evaluation of the platform by showcasing several anomaly detection scenarios based on supervised machine learning classification.

The remainder of this document is structured as follows. Section 2 provides a review of literature on the Intrusion and Anomaly detection topic, including machine learning approaches, event processing techniques and larger, integrated frameworks. Section 3 describes the proposed IADS framework. Section 4 presents several use cases for evaluating the functionally and performance of the framework, and finally, Section 5 concludes the paper and discusses future work.

*Corresponding author
lmrosa@dei.uc.pt (L. Rosa)
ORCID(s): 0000-0002-8230-4045 (L. Rosa); 0000-0001-9278-6503 (T. Cruz); 0000-0002-4939-1773 (M.B.d. Freitas); 0000-0003-1004-6574 (P. Quitério); 0000-0001-7380-9511 (J. Henriques); 0000-0001-7558-2330 (F. Caldeira); 0000-0003-1615-2925 (E. Monteiro); 0000-0002-5079-8327 (P. Simões)

2. Related Work

There are several open challenges threatening SCADA-based Industrial and Automation controls systems (IACS).

First of all, there are dozens of standards, guidelines and best practices recommendations [40]. This creates a fragmentation issue and a challenge to implement them consistently across the entire heterogeneous SCADA ecosystem, from energy related systems (e.g. Smart Grids) to Manufacturing Execution Systems (MES).

The variety of different SCADA communication protocols demands a huge effort, not just to somehow accommodate them all, but also to create solutions able to fit hybrid environments (e.g. multiple SCADA communication protocols in the same domain) and unified solutions that may be reused across different domains. Moreover, most of these protocols still lack fundamental security properties such as confidentiality, integrity, authentication or authorization, leaving room for simple scanning, eavesdropping, replay and data injection attacks. In the last years, several security studies and tools exploring SCADA vulnerabilities were released [78] [57], raising awareness, helping other researches and pushing manufacturers to adopt more secure practices.

Moreover, as we move towards Industry 4.0, IACS become more complex, distributed and data-centric, gradually turning intrusion and anomaly detection into a Big Data problem. While this may be debatable, since there is no significant research to conclude how big is the data in such scenarios (or how big it needs to be in order to be considered Big Data), using Big Data like approaches in IACS security frameworks helps to cope with (1) volume - the amount of information produced by all the interconnected devices; (2) velocity - how to handle real-time information from the physical processes; (3) and variety - how to handle all the heterogeneous information (e.g. sensors data, network traffic, logs).

Finally, the characteristics of IACSs differ from traditional IT environments [45]. IACS operators are known to prioritize availability over security properties. They have typically longer lifetime cycles and fewer updates - exposing SCADA components to known vulnerabilities for longer periods of time. The network communications are also typically more periodic (e.g. repeated polling of sensors data) and have more strict latency requirements.

2.1. Intrusion and Anomaly Detection based on Machine Learning

Intrusion and anomaly detection in IACS has been the focus of several research efforts over the last years. Two main approaches can be found in the literature: signature-based Network Intrusion Detection Systems focused on mainstream SCADA communication protocols (e.g. Modbus, DNP3, CIP, etc.) and anomaly detection based on machine learning algorithms. Signature-based approaches are used to detect abnormal communications patterns based on known packet signatures. They are often not aware of the physical process and are likely to fail against unknown vulnerabilities. On the other hand, anomaly detection based on supervised machine learning models requires previous training and are usually tuned for a single scenario, either for a specific process or for a single communication protocol. Unsupervised approaches are also available but are typically less accurate, which might turn into a huge amount of false positives that overwhelm the operator.

Multiple open-source NIDS, like Snort, Suricata or Bro, have been the focus of several researches to improve their support for numerous SCADA protocols, either by dedicated preprocessors or specific rules [87] [85] [86] [58] [79]. Such signature-based approaches represent an efficient and simple solution for detecting and enforcing communication policies at the network level.

In the remainder of this section, we present an overview of the last four years of theoretical anomaly detection approaches targeting IACS, practical frameworks, and Security information and event management (SIEM) systems.

Phillips et al. [73] presented four different anomaly detection approaches – SVM, Decision trees, KNN, and k-means – to classify SCADA network traffic, having obtained better results with the supervised methods. Similarly to others, their evaluation were based on a public dataset from a gas pipeline [64]. Nevertheless, no details were provided regarding the algorithms implementations nor how such approach could be deployed on real systems to detect attacks on real-time.

McKinnon et al. [59] provided a comparison of three algorithms – One-Class Support Vector Machine (OCVM), Isolation Forests (IF) and Elliptical Envelope (EE) (EE) – for wind turbine fault diagnosis, based on real historical turbines data in Europe. They have obtained the best results (82% of accuracy) with the first two methods. Nevertheless, again, no details were provided regarding the algorithm implementations or the used tools. Moreover, no dataset was available to reproduce the results.

Gao et al. [37] presented an ensemble approach of Feed-forward neural network (FNN) and Long short-term memory (LSTM), having obtained better results to detect temporally uncorrelated attacks with FNN and temporally correlated attacks with LSTM. Among the list of features used, the authors have chosen to include several Modbus-specific[62] fields, thus limiting their approach to scenarios based on this protocol. Moreover, the authors conducted their experiments within a simulated SCADA environment and no datasets are available.

Similarly to the previously described approach, in [88] the features are extracted from network packets into a 25-tuple for DNP3 [28] communications. The authors used a Convolutional neural network (CNN) to classify different types of network-based anomalies into several classes. This has the advantage of not only reporting the anomalies but also classifying them into more specific type of attacks. Even though some of the classes were misclassified, they obtained an overall accuracy of 99.38% and a low number of false positives related to the normal class. The authors assessed their model against two different testbeds, but none of the datasets is available.

A comparison between Support Vector Machine (SVM) and Random Forests against two datasets - Modbus and OPC UA [71] communications respectively - is presented in [8]. The authors identified the most relevant features in the first dataset as a combination of process specific values (e.g. pressure) and packet related features (e.g. packet length). Random forests consistently outperformed the SVM approach. Similarly to most of the SCADA datasets, they started with an unbalanced dataset (i.e. the number of normal values is significantly superior to the anomalies) and a large percentage of missing values. To overcome this, the authors used Principal Component Analysis (PCA) as a preprocessing means for Random Forests and a zero mean scaling for the SVM case.

The performance of two Recurrent Neural Network (RNN) architectures (Gated Recurrent Unit (GRU) and LSTM) is analysed in [81], arguing their benefits against predicting unseen anomalies over the traditional supervised classifiers, more suited to detect known anomalies. Similar to other research works, the authors used a public gas pipeline dataset [64] and a combination of packet and process features. While the authors suggest there is room for improvement regarding the obtained results, the accuracy around 90% for both methods is smaller than other classification approaches. Although more computing-intensive, they obtained slightly better results as the number of epochs increases using LSTM, suggesting it is more suitable than GRU for larger datasets and unlimited computation resources.

Another study [74], follows a different approach, adopting an ensemble technique using a Quadratic Discriminant Analysis (QDA) to combine two density-based estimation algorithms (Local Outlier Factor (LOF) and Subspace outlier degree (SOD)). Such combination avoids the traditional classification path and does not make an assumption about the distribution of the data (as opposed to other parametric methods). Combining both methods, their approach was able to cope with both local and global outlier detection, leading to an overall good performance without penalty for high dimensional data. Nevertheless, this is a computationally expensive method and might not be as accurate as other parametric approaches if the data distribution is known. The authors run their model against the BATADAL dataset [83], allowing to perform comparisons with publicly available results using the same data. The ensemble approach does not always outperform their counterparts and, therefore, one cannot conclude it would always be a better option. Nevertheless, it would be interesting to see how such a technique perform with other datasets and IACS domains.

In another study [51], the authors used a two-level approach by first applying a Blooming Filter and, afterwards, a K-nearest neighbor (KNN) classifier for network anomaly detection. Each packet needs to pass both algorithms to be considered as normal. They used a public SCADA dataset [64] referring to Modbus communications within a gas pipeline facility. The authors handle the unbalanced dataset problem by under-sampling using an AllKNN algorithm. Moreover, the authors also highlight the importance of a feature preprocessing step, having used three different algorithms for that purpose: PCA, Canonical Correlation Analysis (CCA), and Independent Component Analysis (ICA). Nevertheless, and despite their interesting 97% of accuracy, they obtained significantly different results (from 68% to 100%) depending on the class of anomalies. Moreover, since they depart from Modbus-related features, it would be interesting to understand how they perform on different protocols and feature spaces.

Another interesting approach with 98% of accuracy combined Online Sequential Extreme Learning Machine (OS-ELM) with a set of Restricted Boltzmann machine (RBM) to classify network flows into several classes of attacks [24]. It also used the public dataset of a Gas pipeline [64] but does not specify whether the entire feature space is used or not, only mentioning that the data is grouped into sliding windows of 100 samples with overlapping of 400 instances. OS-ELM is used as a first step to classify the flow as normal or in one of the known classes. Whenever an anomaly is detected, it is forwarded to one of the RBMs. Then, each RBM, trained for a single class of anomalies using a unary classification method, is responsible for deciding whether the anomaly matches its class or not. If there is no match it will sequentially test the remaining RBMs. Such extra step might be valuable to improve the correct class classification since its RBMs can be highly specialized. Nevertheless, it is important to acknowledge that the OS-ELMs are susceptible to produce false negatives. In that case, the flow is marked as normal and never reaches any of the RBMs. Moreover, this approach depends on the previous knowledge of each class and might not be suitable for unknown types of attacks.

A three-stacked LSTM approach to predict anomalies in time series windows against different types of datasets, including one power demand dataset, is proposed in [67]. Despite claiming 92% of precision, no details are provided about the used features or the used datasets (which are not publicly available).

An approach to predict cyber-attacks by analysing different combinations of CNNs is presented in [53]. It successfully detected 32 out of 36 attacks in a public industrial water treatment dataset [41], processing physical process parameters in time windows of 200 seconds. Nevertheless, using only physical process parameters means this approach will not detect layer 2/3 attacks without direct impact on the physical process (e.g. network scans).

Intrusion Weighted Particle based Cuckoo Search Optimization (IWP-CSP) and Hierarchical Neuron Neuron Architecture based Neural Network (HNA-NN) are used to classify SCADA network data into nine different classes in [80]. The authors mention an experimental evaluation using a simulated environment, with 100 nodes and two process level features (i.e. humidity and temperature). However, there is no information on how such a small number of features are related with their feature optimization layer, how they relate with all the categories of attacks, or whether an additional set of features were used but not referred.

The authors of [50] present an anomaly detection solu-

tion capable of detecting attacks against managed physical processes based on: 1) a first stage using a SVM model to detect whether there is an anomaly and 2) and a second set of SVMs specifically trained to classify the anomaly into a known category. They evaluate their model against a Human in the Loop (HITL) testbed, using a hybrid combination of a simulated process (Tennessee Eastman chemical process) and a real PLC (Wago 750-881). Time windows are used to minimize data noise and to reduce the number of false positives. The model was able to detect the two injected payloads. Nevertheless, it requires training and it will likely fail to detect other categories of attacks that do not interfere with process values.

Table 1 presents a summary of the surveyed literature. Since each research work might contain multiple experiment variants, the presented performance indicators refer to the best results (scenarios) found for each work. Similarly, since multiple distinct datasets are often used during the evaluation step, the presented results prioritize SCADA-based validations. Additional surveys in the topic of SCADA-anomaly detection, covering previous works, can be found in [26] [66] [45]. Additional anomaly detection literature focused on more general cybersecurity scenarios can be found in [29] [42] [68] [84] [25] [5].

2.2. Security Information and Event Management Systems for SCADA

Opposed to the previous section, that only addressed machine learning approaches, this section discusses related work towards more complete frameworks that focus on, among others: (1) collecting information from several sources in real-time; (2) having the capability to preprocess and aggregate those events; and (3) processing those events and deciding whether they are anomalies or not.

A tri-modular platform to support existing cybersecurity tools at Smart Grid Control Centers, composed of a data module, a classification module and an action module, is presented in [82]. This is an interesting approach that explores the advantages of having several types of data sources and a data layer that implements several types of data preprocessing and data ingestion. Nevertheless, the analytic component is limited by a single classification approach (i.e. LSTM), which might be not sufficient to cope with all types of anomalies. Moreover, details about the internal parts of the framework are not provided. This raises multiple questions, such as: How the outputs of several heterogeneous sources are jointly processed? Which data format is used? What feature space is used in the classification module?

Similarly, the authors of [52] proposed another interesting framework for fault detection, targeting oil and gas industries. The authors proposed a lambda architecture based on widely used open-source tools (i.e. Apache Kafka, Spark and Cassandra). Nevertheless, few details about the implementation and obtained results are provided.

The authors of [89] focused on time-series analysis within an Industrial field environment, presenting a Big Data framework that encompasses several layers: acquisition, transmission, data processing and visualisation. All components were based on widely used open-source tools such as MQTT, Apache Kafka, Spark, InfluxDB and Grafana. A custom JSON-based message format was used to seamless communication between the components. Despite the potential value of such a platform, there are no implementation details or more practical evaluation results.

A SCADA honeypot environment together with a SIEM system for cyber-attack profiling is presented in [56]. Although the platform was not developed for detecting cyber-attacks within a real SCADA environment, it provides some related functionalities, such as log processing capabilities, a real-time processing approach and a dedicated visualization interface. A critical point for a good intrusion and anomaly detection tool is to avoid replacing all the existing and specialized detection components such as NIDS by a global anomaly detecting based on a given machine learning model. Instead, it should be able to add knowledge on the top of the existing technologies. Similarly to others, the authors designed the platform based on open-source software such as an Elasticsearch stack [31] and Suricata [72]. Nevertheless, no details were provided about the used visualization techniques, the evaluated processing algorithms or the adopted message formats.

In [47], the authors explored the Suricata detection plugin functionality to perform an additional packet inspection to recreate a stateful analysis of the IEC 61850 [2] protocol family. This is an interesting approach to complement single packet signature matching, especially for complex protocols.

The desired features and challenges of a policy-based SIEM are analysed in [38]. This is a different but important approach, focused on the correlation of general business policies rather than on physical process values or network traffic. The authors describe the classical example of a logon into a system event, detected after a leaving event. Individually, each of the events could be regarded as normal, but when considering their sequential order they reveal an anomalous pattern.

A framework focused on EtherNet/IP and CIP [70] protocols is presented in [39]. The authors extended the functionality of Bro, an open-source NIDS, and proposed a hierarchical deployment of several instances (based on IEC 62443 SCADA reference architecture levels [3]). All the Bro logs were aggregated using ElasticSearch and visualized using a dedicated web interface. While this approach can successfully detect several types of attacks, few implementation details were provided regarding the framework itself.

Table 2 presents a summary of the surveyed frameworks. Most of them were designed to address a single specific problem or a single protocol. The ability to combine multiple sources, techniques and protocols into a comprehensive framework for supporting SCADA operators' decisions seems to remain a challenge.

2.3. Discussion

Before diving into a specific approach, it is important to first understand the problem.

Table 1
Summary of the latest literature contributions of SCADA Anomaly Detection Algorithms

Ref.	Year	Focus	Method	Dataset	Features	Anomalies	Indicators
[73]	2020	Gas Pipeline	SVM, Decision trees, KNN, and k-means	Public Dataset	Process and network related	L7 Attacks	ACC=99.99
[59]	2020	Wind Turbine Systems	OCSVM, IF and EE	Custom Dataset	19 Turbine features	Historical turbine faults	ACC=82
[37]	2019	Industrial Control network data (Modbus related)	FNN and LSTM	Custom Modbus Related	19 Network Packet Features	Layer 2/3/7 attacks	P=99.76 R=99.57 F1=99.68
[88]	2019	Industrial Control network data (DNP3 related)	CNN	2 private testbeds	25 Packet Based	Layer 2/3/7 attacks	ACC=99.38
[20]	2019	Industrial Control network data	AMPSO-SVM-K-means++ / GSA-AFSA-ELM	N.A.	4 Network packet related Features	N.A.	DR=95 FA=0.02
[8]	2019	Gas pipeline (Modbus and OPC-UA related)	SVM and Random Forests	2 Public data sets	Network packet and application based	Layer 2/3/7 attacks	ACC=99.98
[81]	2019	Gas Pipeline	RNN (LSTM and GRU)	Public Dataset	N.A.	35 application level attacks	P=0.92 R=0.92 ACC=0.92
[15]	2019	Electric Power Systems	CNN	Simulated IEEE-Bus systems	N.A.	Power related faults and Data injection	ACC=98.67
[74]	2019	Water distribution	SOD / LOF / QDA	BATADAL	N.A.	Application Level Anomalies and Replay attacks	P=0.88 R=0.94 F=0.91
[51]	2019	Gas Pipeline	Bloom Filter and KNN	Public Dataset	20 Network Packet Features	Layer 3/7 Attacks	ACC=0.97 P=0.98 R=0.92 F=0.95
[24]	2019	Gas Pipeline	OS-ELM and RBM	Private dataset	26 attributes	Layer 3/7 attacks	P=0.99 R=0.99 F=0.99
[53]	2018	Water treatment system	CNN and LSTM	SWaT	51 attributes	36 application level attacks	P=1 R=0.85 F1=0.92
[67]	2018	Power Demand	LSTM	N.A.	N.A.	Power Demand faults	P=0.92 R=0.14 F1=0.87
[46]	2017	Electric Power Systems	SVM / ANN	Transmis-sion & Distribution datasets	5 PMU correlation Features	Replay Attacks	ACC=98.47 P=99.54 F1=0.92
[80]	2017	Sensors Data	HNA-NN	100 nodes simulated in NS-2	2 Features (humidity and temperature)	DoS and Spoofing attacks	P=72 R=100 ACC=95
[50]	2017	TE Chemical Process	SVM	Simulated TE process	12 sensors measurements	Ladder Logic Injection	Graphic only

Precision (P), Recall (R), F-Score (F), Accuracy (ACC), Not Available (N.A.)

Table 2
Summary of the latest literature contributions of SIEM related Frameworks

Ref.	Year	Focus	Data Sources	Data Format	Messaging	Analytics
[82]	2018	Smart Grid	Logs and Sensor data	N.A.	Kafka	Spark, R and TensorFlow
[52]	2017	Oil and Gas Industry	Logs and Sensor data	N.A.	Kafka	Fault detection using ESPER and Spark
[89]	2017	Industry 4.0	OPC / Network Data	Custom JSON based	MQTT / Kafka	Storm
[56]	2016	SCADA network traffic	Public exposed SCADA Honeypot	Suricata alarms	N.A.	Profiling analysis using ELK stack
[47]	2016	Smart Grid	IEC 61850 Network Traffic	Suricata alarms	N.A.	IEC 61850 Stateful protocol analysis based on Suricata Plugins
[38]	2016	Policy Monitoring	Security Control Logs	Custom Structured Logs messages	N.A	Log Correlation
[39]	2016	Water Treatment	CIP/EthernetIP traffic	Bro alarms	N.A.	Log aggregation from multiple domains with ELK

The topic of SCADA security is a large umbrella for different specific problems. The majority of the reviewed papers focused on detecting anomalies such as network-based cyber-attacks or physical faults by looking either at physical process properties, network SCADA communications or a combination of both. Other potentially relevant features, such as diversified log sources and host-based events are commonly ignored, therefore missing an important opportunity to develop a more comprehensive approach covering a broader spectrum of attacks. It is very unlikely that a single approach would outperform all the others for all types of anomalies. For instance, a model considering process features only will simply fail to detect an important range of attacks that do not have a direct impact on the process, such as network scans or brute force login attempts. If we look at the example of recent APT campaigns targeting IACS [48], they typically start by collecting information from the environment over long periods of time - a latent stage during which the threat may go undetected for months if one just looks at physical process values. A NIDS (with support for SCADA protocols) is still a valuable source for detecting such kind of unauthorized communications - one of the most referred SCADA security issues. Nevertheless, since a NIDS is limited to network traffic, it is clearly insufficient to address all the security needs. This suggests that a better strategy is to include more data sources.

On the other hand, the complexity of a SCADA environment is one of the biggest challenges for anomaly detection based on machine learning approaches. The amount of potentially different types of features can severely affect the performance of the model and should be carefully selected. Another frequently observed problem is a lack of datasets. Given the cost and the complexity of recreating an industrial control system, most surveyed works focused either on small scenarios or in the few publicly available datasets. Moreover, some of them lack a diversity of anomalies, resulting in imbalanced datasets that might lead to inaccurate or biased classification issues. Intuition also suggests that a better approach is to take into consideration both supervised (more apt at detecting known issues) and unsupervised (more suitable to detect unknown anomalies) techniques.

Similarly, by combining several different approaches, it is possible to infer both local and global anomalies (e.g. a single component fault on a given domain might not have immediate effects on the overall process). Additionally, other techniques such as stream processing (especially windowing) might help to reduce the number of false positives, since the analysis is performed on the top of a group of events, rather than a single event. Finally, as we are moving towards a Big Data problem, to avoid scaling issues it is critical that the chosen approaches can fit into scalable, distributed and parallel computation environments.

3. Proposed IADS Framework

In this section, we discuss the design of an intrusion and anomaly detection framework capable of coping with the next generation of IIoT-centric IACS systems, which are expected to be highly distributed and capillary (as it is the case for smart grids, for example). This evolution is expected to bring new security challenges, namely: larger attack surfaces, often coupled with insecure communications protocols; likely insecure components in the long term, due to long life-cycles; and increased complexity, both at the process level and business levels. On the other hand, at the architectural level, there are challenges such as the increasing volumes of heterogeneous data, the event processing schemes, the elasticity of the platform, the need for interoperability between components, the deployment approaches and the overall platform orchestration.

The main idea behind our proposal is not to deploy a single detection mechanism, but rather to have a flexible framework and a holistic SIEM capable of collecting valuable data from multiple domains, efficiently processing that data, and integrating multiple detection techniques that, combined, provide more knowledge about the security state of an infrastructure. The combination of different probes and third-party components also has another interesting positive side effect: it helps to cover a larger attack surface and makes it more difficult for an attacker to bypass multiple detection mechanisms.

The remaining of this section presents the building blocks of the proposed IADS architecture (see Figure 1), including a set of distributed probes that can be deployed on-demand to meet different requirements, an elastic data streaming platform that can easily adapt to different topologies, and a global event processing framework that integrates multiple techniques.

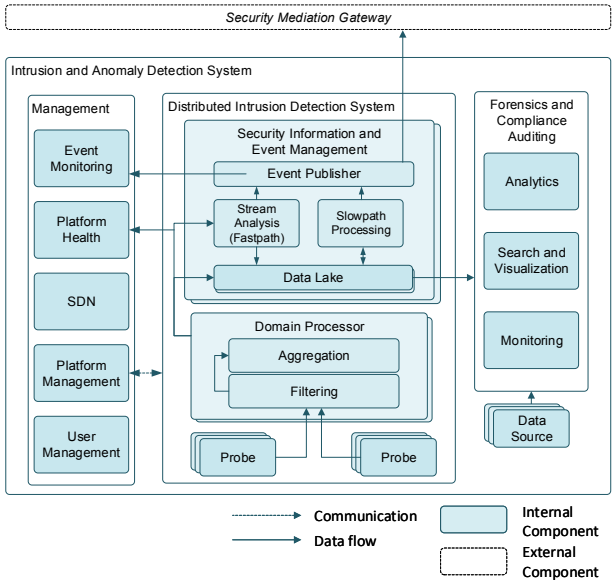

Figure 1: Intrusion and Anomaly Detection System architecture

A security monitoring solution for the next-generation IACS must focus on the quality and the timing of the data. More than ever, there is a multitude of event sources (e.g. field data, network traffic, infrastructure health information) that need to be monitored in near real-time. Our IADS was designed as a data-driven security framework to fulfil those requirements, combining different types of components (encompassing custom developed modules, as well as existing open-source tools) with advanced event processing strategies (e.g. stream processing and batch processing) that can be used to efficiently collect and process different types of data.

Another important thing to refer is that not all the IACS environments and domains have the same requirements. There is no magic number or metric to anticipate how much relevant data needs to be collected and processed, for security monitoring purposes. For instance, according to [32], even a small country like Portugal has 6.5 million electricity metering points. Whereas it is not expected that a single solution would be used to monitor them all, such number does not account for all the other sources of information such as security-related probes, network traffic or application logs. So, depending on the measuring rate and the actual scenario, we might expect from a few dozens of events per second (e.g. small electricity substation) up to millions per second for a large deployment. Therefore, the flexibility of the platform and the potential to scale in/-out to accommodate different deployments are also key requirements.

3.1. Cyber-physical probe and component management

Cyber-physical probes are also an important part of the platform. They are strategically deployed components, either physical or virtual, used to continuously collect field evidence, from raw telemetry data (e.g. physical process-related information) up to complete reports of a malicious activity incident.

Since the lack of security on SCADA communication protocols is one of the most criticized topics of these environments, we focused on understanding the existing SCADA support for widely used open-source Network Intrusion Detection Systems (NIDS)s (i.e. Snort, Suricata and Zeek). Although all of them have some SCADA support, there is still plenty of room for improvement with additional protocol support, more rules and enhanced functionalities. We contributed with multiple SCADA signatures for different protocols [75] [27].

In the context of our architecture, it was also important to understand how to integrate each probe output into a common data format. There is a lack of a universally accepted format. Each tool tends to use its own custom format or inefficient formats such as IDMEF [33], a heavy format based on XML which is not suitable for Big Data approaches. Nevertheless, the event format is a key aspect, from both interoperability and efficiency standpoints. A common format allows seamless integration of third-party components. Whereas, choosing an efficient format have a direct impact on the des-/serialization tasks, the network transmission and the event processing algorithms. Taking this into account, we designed a generic data-model for describing both incidents and telemetry information [79], based on the Apache Avro serialization system [9].

We also developed a common event adaptor and a generic configuration agent for all the probes and components [79]. The event adaptor, based on a YAML configuration file and a set of regular expressions, was responsible for mapping different output formats and log sources (e.g. Syslog) into a common event format (i.e. our data-model) and then for pushing those events into upper layers.

The configuration agent, built on top of the MQTT protocol [7], provided a unified approach to manage different components by abstracting their settings into a text-based configuration and a set of common actions (read, write and

exec). This allows to remotely and dynamically reconfigure each component property through a common web interface.

Additionally, we explored SDN / NFV technologies to implement the virtual probe concept, designed to leverage the elastic nature of the platform. We also used a component registry to support a virtual probe template library, allowing for different types of instances to be deployed on-demand, in a matter of seconds [35]. Finally, we developed physical and SCADA specialized probes such as the Shadow Security Unity (SSU) [23], as well as additional instrumentation probes (refer to evaluation section for details) to further enhance the data collection stage.

3.2. Event stream processing

The event streaming platform plays a key role in the IADS architecture, fulfilling two purposes: (1) to provide an efficient, distributed and decoupled mechanism for inter-component communication with *exactly-once* processing guarantees; and (2), to provide domain-level processing capabilities. The latter is particularly suited for taking advantage of edge computing capabilities, enabling a distributed preprocessing stage closer to the event source. For an anomaly detection platform, this is useful for optimizing the entire processing pipeline, enabling all sorts of event correlation techniques (from simple event filtering up to event enrichment and event aggregation) at an earlier stage of the processing pipeline.

We used Apache Kafka [10] for implementing both the messaging system the and domain processors. This enabled a flexible and tight integration between both. This integration allows a unified approach to keep up with the new distributed boundaries of next-generation IACS, supporting both small and large widespread deployments, as well as per-domain edge computing capabilities.

Moreover, Kafka was designed to achieve high message throughput without sacrificing latency – potentially supporting millions of messages per second. Additionally, Kafka has multiple fault-tolerance and durability capabilities. A message can be replicated across multiple Kafka topics, with each topic possibly partitioned across several Kafka brokers. This flexibility constitutes a key advantage for supporting different message requirements within the same deployment (e.g. high-severity messages vs low-priority telemetry data).

Another useful Kafka concept, opposed to pure queuing mechanisms, is that the messages are not deleted after being consumed. Instead, Kafka uses a sequential offset to identify each record within a partition and has the notion of consumer groups. Moreover, a message can be load-balanced across different consumer-members and multiple consumer groups can consume the same message. For instance, at the same time, we can have multiple instances cooperatively aggregating the events, and we can also have an off-the-path task to persist all of them (e.g. via Kafka Connect API to the Data Lake).

The native support for stream processing is another advantage of Kafka. In the IADS, multiple domain processors, built on top of the Kafka Streams API, can be dynamically deployed, on-demand, to support different preprocessing requirements (e.g. different message priorities) on a per (logical and physical) domain basis.

Each domain processor is composed of a topology including a source (the topics where the probes produce) and a sink (the input topics of the SIEM layer). Additional intermediate topics can also be used for supporting stateful operations (e.g aggregations), in contrast to stateless operations (please refer to Kafka documentation [14] for such distinction).

Figure 2 illustrates an example of how we leverage Kafka Streams DSL (Domain Specific Language) [14] to implement a feature aggregation task. Such tasks are the foundation of the domain processor concept. In this example, we implemented a domain processor to extract additional aggregated features grouped by time windows on the top of a stream of individual network packet features.

The initial `KStream <K, V>` is instantiated from one or more input topics, where `V` refers to the IADS messages. Next, we map all the messages to a common key across all the messages, to perform a global aggregation. This is followed by a further split of the aggregated stream (a `KGroupedStream`) into a time windows feed (a `TimeWindowedKStream`). Additional aggregation scenarios are possible, using a different `K` mapping in the previous step (e.g. using the `uuid` field in the IADS datamodel for unique counts, the `origin` field for further aggregation by message sources or the `severity` field to distinguish between different message priorities). Similarly, different time windows types are supported through the Kafka Streams API (e.g. tumbling, hopping, sliding, etc.).

Then, each individual feature, encoded within the IADS message using a list of *meaning*, *content* field pairs, is extracted into a new aggregated message. The output (the computation result of the aggregation) is continuously stored on a per-event basis. In our scenario (cf. Section 4 for more details) we use one-pass algorithms (e.g. Weldford's algorithm for computing the variance) to optimize the computation of the aggregated features. Similarly to the original messages, the aggregated messages contain the aggregated features encoded using a list of *meaning* and *content* field pairs.

Finally, the aggregation result is transformed back to a plain `KStream<K, V>` (containing the final aggregated stream of messages) and ultimately pushed to the sink topic. It should be noted that the Kafka Streams API supports different types of windows behaviours. For instance, each message in the input topic might turn into an output message, where each output message represents an intermediate result of the aggregation (default behaviour). In our implementation, we use the `suppress` option to produce a single message, containing the final aggregation, per each time windows. Moreover, the time windows might by event-driven, using the event time of each message, or triggered by the wall-clock time using schedulers.

3.3. The SIEM Component

The SIEM component aggregates the event streams from the multiple domains and, based on a set of machine-learning algorithms, ultimately classifies an event as normal or anoma-

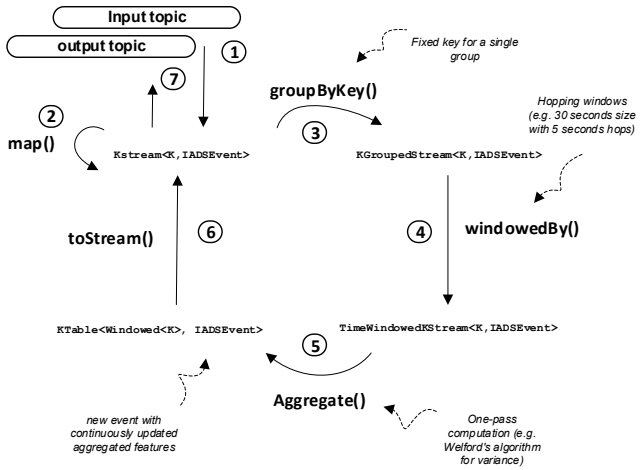

Figure 2: Feature aggregation by time windows using Kafka Streams DSL API

lous. We use Apache Spark [11] to reassemble a lambda architecture in order to have a unified platform capable of handling two types of data processing techniques: stream processing (fast path) and batch processing (slow path).

Spark is a distributed data processing framework that provides significant improvements over traditional map-reduce alternatives, by using an in-memory approach and an efficient Directed Acyclic Graph (DAG) scheduler. Similarly to Apache Kafka, the distributed nature of Spark and the notion of workers and executors provide an extremely elastic approach where multiple computation nodes can work cooperatively. Moreover, it already contains a set of APIs for structured streaming, machine learning and dataframe manipulation. Factors such as the native Kafka integration (allowing the usage of Kafka as a source and sink), the Avro support and the availability of third-party machine learning algorithms implementations were also useful. For the IADS, this means it is possible to natively implement a set of complete machine learning pipelines directly on top of the event streams coming from multiple domains.

Figure 3 illustrates an example of an anomaly detection task used to classify each message from a Kafka topic (with the messages produced by the domain processors containing a list of features) as normal or anomalous. It showcases the integration between Kafka and Spark – and how the different Spark APIs can be leveraged in the context of a SCADA anomaly detection. The prediction step requires a classifier, previous-trained based on the collected datasets (cf. Section 4 for more details).

A *Kafka Source* is configured to subscribe to multiple domain level topics using the Spark Structured Streaming API, which supports different types of input sources. Each Kafka message (`K V` pair) coming from the domain processors is directly mapped into a `Row` of an unbounded Spark `Dataset<Row>`. The `Row` contains a list of columns including, amongst others, both `K` and `V` encoded as binary values, as well as the topic and partition metadata.

Plain Avro messages can be directly decoded at this step to primitive and complex Spark column types, based on the Avro schema. For a common Avro variant used by Confluent Kafka Registry [1], where each message contains also the schema management information, an extra conversion step was necessary. Additionally, we also used a custom lookup function for converting known feature pairs (based on *meaning* / *content* field entries) and saving them as an additional `Vector` column of the initial `Dataset<Row>`.

Then, by leveraging the Spark ML API, each feature vector goes through a pipeline, meaning a chain of *transformers* (e.g. one-hot-encoding) and a *estimator* (the classification algorithm). The classification result is also stored into a *label* column. Finally, for each detected anomaly, a new message is derived and pushed back to a Kafka topic using a *Kafka sink*.

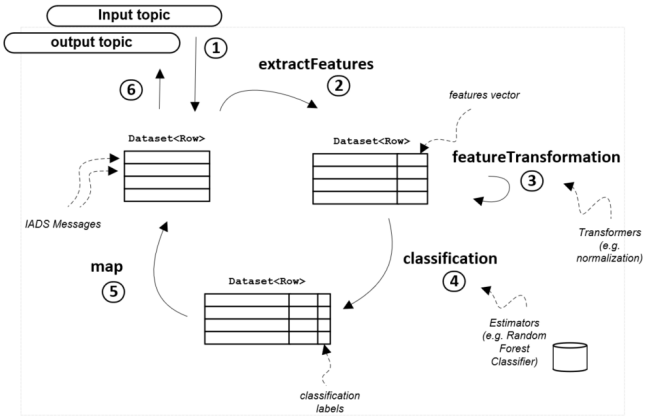

Figure 3: Overview of a classification task using the different Spark APIs

3.4. Other Components

In this section we briefly address two relevant components of the proposed IADS that, despite not being extensively explored in this paper, distinguish our proposal, thus deserving a mention.

One of these components is the leveraging of Software-Defined Networking (SDN) and Network Function Virtualization (NFV) technologies into the IADS. This allowed having different types of containerized probes and services, including virtual NIDS [35], by on-demand assigning them monitoring ports or virtual data-diodes that are cost-efficient alternatives to enforce unidirectional communications [36]. The SDN controller itself can also be used as an additional source of information by providing soft real-time statistics of network flows and virtual switch ports. The required redesign of the SCADA network might create a barrier for the SDN adoption, in some scenarios, but the numerous benefits we observed [35] clearly pay off.

The other component, orthogonal to the event analysis layer, is a Forensics and Compliance Auditing (FCA) subsystem [43] [44](under revision) for persisting a broader spectrum of digital pieces of evidence obtained from additional data sources such as Authentication, Authorization and Accounting (AAA) sessions or physical access control systems.

This module serves as an assisted mechanism for forensics analysis tasks and policies conformity checks. The FCA module uses the Elasticsearch stack (Elasticsearch, Logstash, and Kibana) [31] as an efficient manual and semi-automatic searching tool to cope with the complexity and massive amount of available data.

4. Evaluation

This section describes several scenarios, the implementation and tools used for evaluating the proposed IADS – with a special focus on the event processing and anomaly detection layers.

First, we present the reference industrial testbed which provided support for our experiments. Next, we describe the evaluation use case that we used to assess the performance of several supervised ML algorithms (using our platform). Finally, we evaluate the event messaging layer, discussing different configuration and service goals scenarios.

4.1. HEDva Testbed

The functional validation was conducted in the HEDva testbed [34], a facility built by the Israel Electric Corporation (IEC) to recreate the various domains of a smart grid, from power generation up to power consumption. The testbed is composed of a hybrid combination of real SCADA components from multiple vendors with physical processes emulated with high detail. This mix makes it possible to assess the vulnerabilities of different SCADA protocols and equipment without worrying about physical damages (as would happen in a production environment).

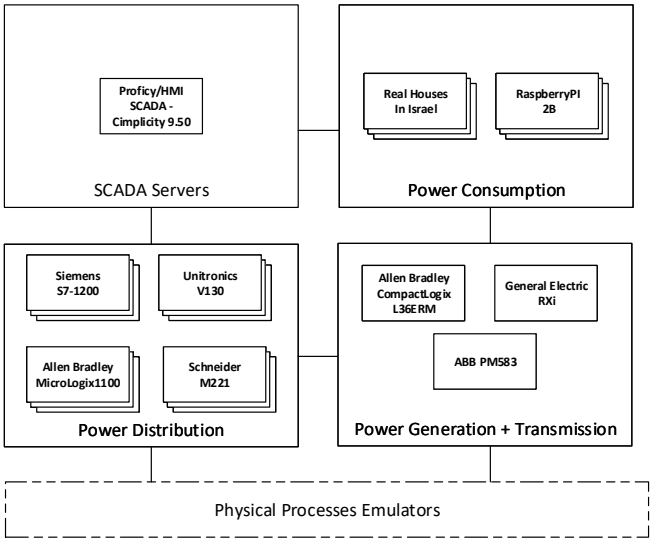

Figure 4: HEDva Testbed Overview

4.2. Reference Scenario – Data Exfiltration

A commonly observed scenario in cyberattacks is the establishment of side channels to the Command and Control servers (C&C). These channels are used to remotely manage the compromised hosts as well as to exfiltrate sensitive data. DNS tunnelling is one of the many known techniques that allow recreating a two-way communication channel by encoding data on both DNS queries (as part of the queried domain name) and DNS responses (in the resource record data). DNS is a fundamental network protocol for several services and, therefore, it is often not blocked by firewalls, leaving room for successful DNS-based exfiltration. Whereas DNS is not a SCADA specific protocol, it is a candidate to be used within SCADA systems to exfiltrate confidential data such as PLC's ladder logic, process data or operators credentials. Various machine learning approaches have been proposed in the literature for detecting DNS-based exfiltration, mainly organized in two approaches: payload analysis (based on packet header fields) and traffic analysis (based on DNS session metrics) [90] [65] [69] [91].

Figure 5 illustrates the setup used to recreate a data exfiltration scenario using DNS tunnels. First, we configured a server in the cloud to behave as an authoritative DNS server for two previously registered domains. Then, we used two popular DNS tunnelling tools (dnscat2 [16] and iodine [30]) to produce 23 scenarios variations, from simple tunnel handshakes using different DNS record types to encrypted sessions, interactive shells and the exfiltration of a complete PLC project. On the IADS side, to showcase the different anomaly detection capabilities, we developed a DNS probe capable of extracting packet-specific features (as described below) from the DNS network traffic. We also developed a domain processor capable of extracting additional features based on the aggregation of the messages into time windows. Finally, a SIEM application was developed for training/classifying each message as normal or anomalous (i.e. a possible attempt of DNS tunnelling).

Three datasets resulted from the experiments (cf. Table 3). Dataset *DS1* contains over 15 thousand records (one per each DNS packet), including all the anomalous and normal DNS traffic related to software updates, network services and arbitrary DNS requests. Dataset *DS2* contains the aggregated features - one record per each time-based window - using hopping windows of 30 seconds and 5 seconds hops. Dataset *DS3* contains 25 features derived from *DS1*, including the average and the standard deviation for each feature on DS1, as well as the number of packets per window.

Despite being originated from the same scenario, the three datasets are fundamentally different and were used to feed different anomaly detection approaches. There are trade-offs between them. In DS1 we depart from a larger dataset where each record, containing individual features of a single network packet, can be processed in near real-time. This means less computation on the preprocessing step but more at the classification algorithm. In DS2, only aggregated events are used to train the anomaly detection system, meaning more computation at the domain processing but fewer events to classify in the upper layers. Moreover, there is a distinction between DS2 and DS3. In DS3 we use DNS-specific aggregated features, whereas in DS2 we pursue a more generic approach (that can be applied to other protocols – including SCADA protocols) by using only generic network-level

Table 3
Number of Records per class and dataset

	No of Records		
Class	DS1	DS2*	DS3*
Normal	8458	868	868
Anomaly	7410	276	276
Total	**15868**	**1144**	**1144**

*Original data before oversampling using SMOTE technique.

features. Whereas the final accuracy in both cases is totally dependent on the chosen features, a single packet might be insufficient to represent a cyber-attack. On the other hand, an aggregated record might be enough to flag a high network pattern but might hide information behind all the underlying aggregation statistics for low profile attacks. In DS3 we use overlapping time windows to reduce the latency between each new record, while maintaining larger aggregation windows.

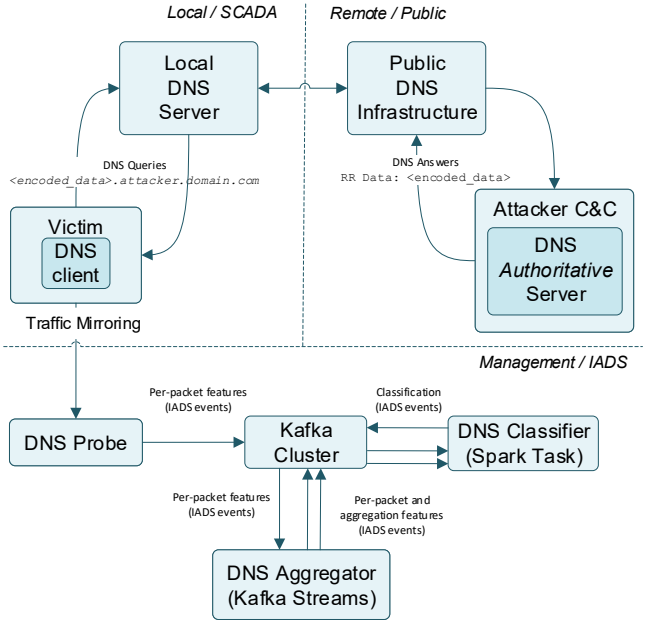

Figure 5: DNS tunneling scenario

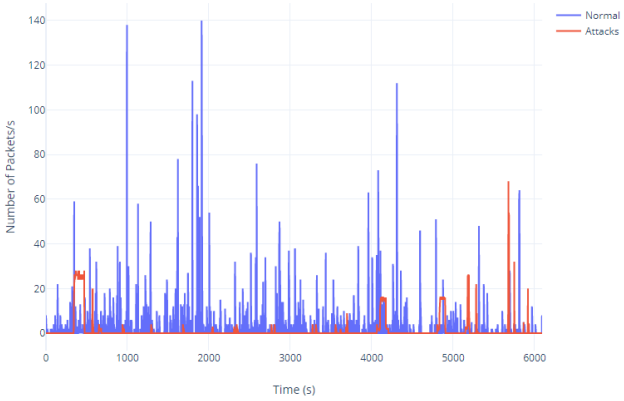

Figure 6: Distribution of packets over the time

Table 4
Features list by dataset

DS1	DS2	DS3
Frame Size	Count	Count
Query length	Frame Size Mean	Mean of each feature of DS1
Query Shannon entropy	Frame Size Std	Std of each feature of DS1
Query char percentage	Frame Size Var	PIT Mean
Query non hex percentage	PIT Mean	PIT Std
Query Alexa top 1M	PIT Std	
Reply length	PIT Var	
Reply Shannon entropy		
Reply char percentage		
Reply non hex percentage		
Reply Alexa top 1M		

Packet Inter-arrival Time (PIT), Standard Deviation (Std), Variance (Var)

For anomaly detection we used a machine learning approach where we evaluated the performance of nine supervised algorithms for binary classification (cf. Table 5). Each record was labeled and marked as anomalous if either the request or response contains one the malicious domain names – therefore, this feature was not considered for any algorithm. The full list of used features, inline with previous works, is enumerated in Table 4. Their feature histograms, depicted in Figures 7, 8 and 9, show their pairwise distribution.

The features are mainly focused on: (1) the size of both request and response packets – DNS tunnelling packets are typically larger because they carry extra information; and (2) how different the domain names are from normal DNS requests – since they are used to encode data, they are typically less natural than words used in normal DNS requests. Additionally, we also include a binary feature to indicate if the

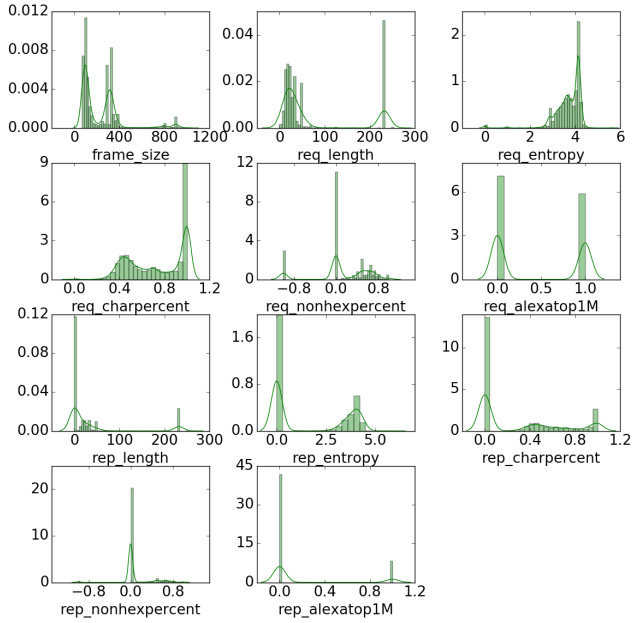

Figure 7: Features Histogram for Dataset DS1

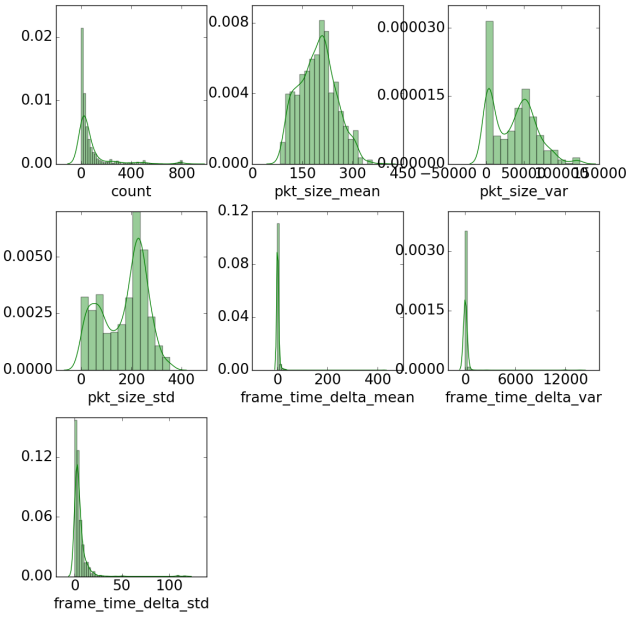

Figure 8: Features Histogram for Dataset DS2

domain is listed in the Alexa top sites list [6], as an indicator of whether this is a common DNS name. Other common features, such as layer 2/3 fields or time-related measurements, were explicitly not used. This means this approach is not constrained to specific network details or limited to high-throughout attacks. Opposed to other research works, we decided to leave out the DNS record type. Although some record types are preferred over others for data exfiltration, using this feature for training purposes can result in inefficient models for classification of all DNS tunnel variants. The dataset we collected contains a combination of several

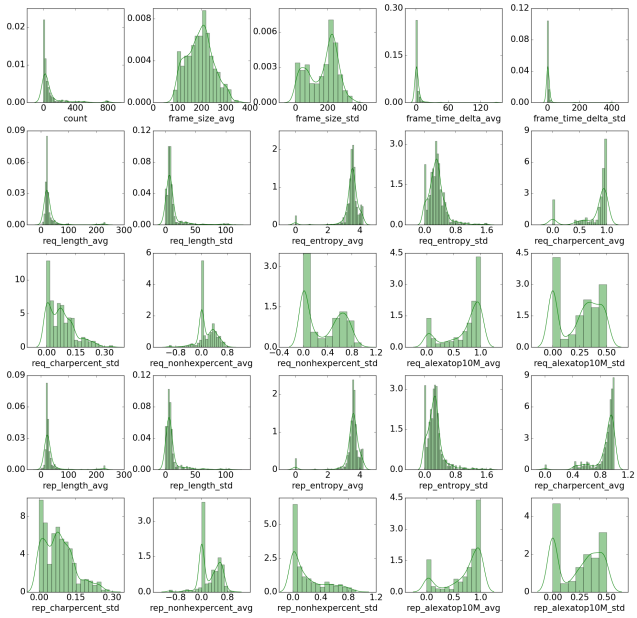

Figure 9: Features Histogram for Dataset DS3

tunnelling samples using different record types. We included features extracted from both queries and replies. Although there are some correlations between these features, since we might have data encoded on both directions or only one way, we opted for not performing any feature reduction, due to the reduced number of items. Figures 10, 11 and 12 show the heat maps of the Person correlation coefficients between the features for each dataset, whereas Figures 13, 14 and 15 show the mutual information between each feature.

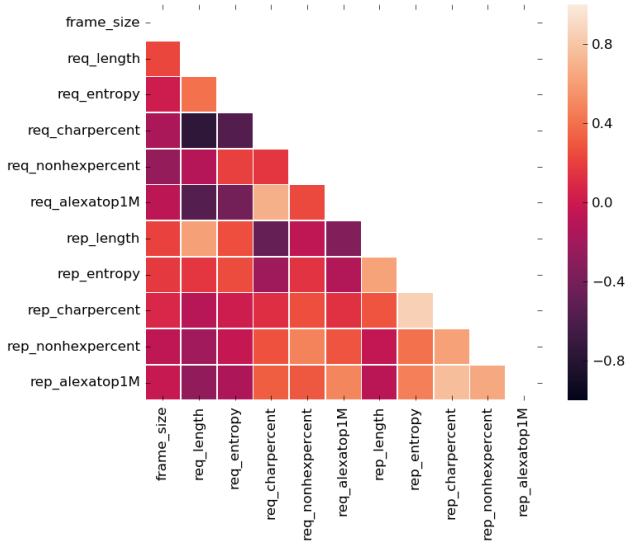

Figure 10: Feature List Person Correlation for DS1

The application was developed on top of most recent dataframe-based Spark ML API (instead of RDD). This allows a more user-friendly approach, by introducing the notion of *pipelines*, a set of *transformers* and *estimators* that

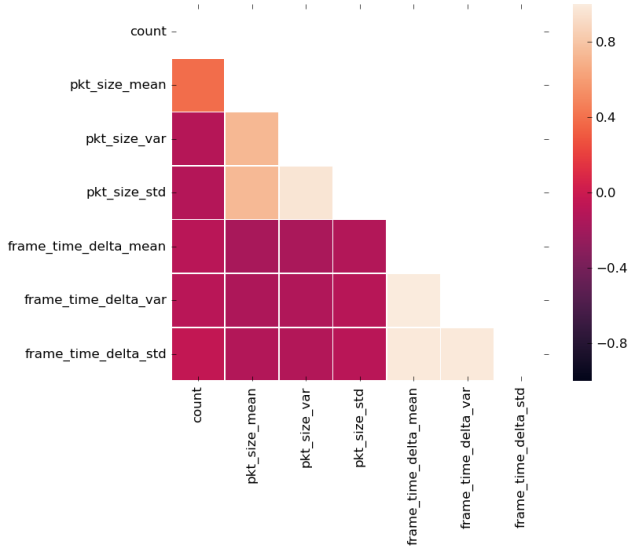

Figure 11: Feature List Person Correlation for DS2

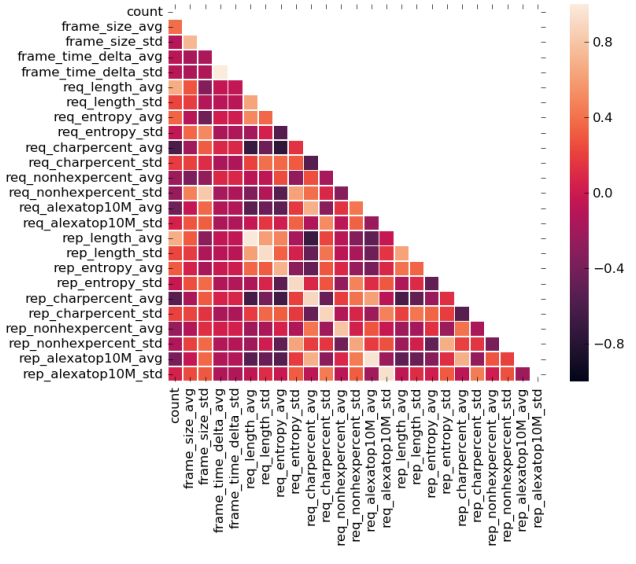

Figure 12: Feature List Person Correlation for DS3

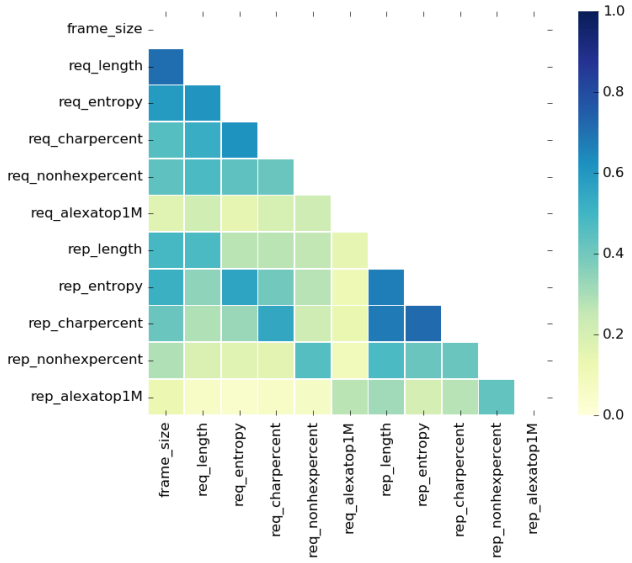

Figure 13: Feature Pairwise Mutual Information for DS1

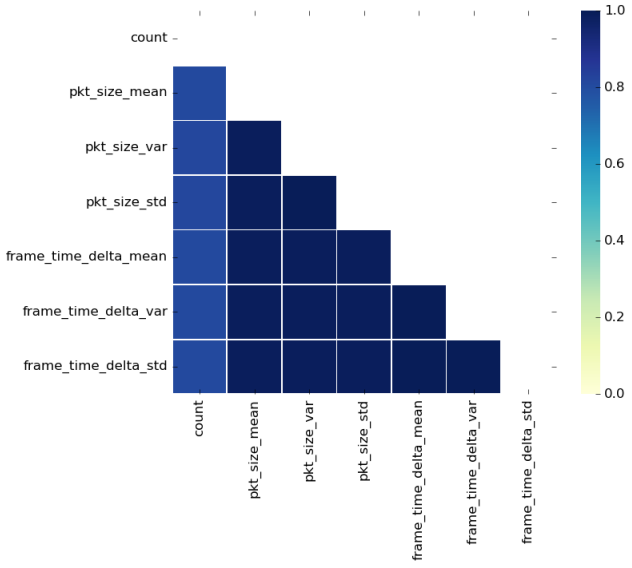

Figure 14: Feature Pairwise Mutual Information for DS2

can be applied directly to a dataframe to reassemble a complete workflow. For all the algorithms, we started from a random 70/30 split where 70% of the dataset is used for training, validation and hyper-parameter tuning and the remaining 30% was used for the final testing. Depending on the specific algorithms (as detailed below), the first step of the pipeline is the feature transformation (e.g. scaling, one-hot encoding, etc.). The next step consisted of a 10 k-fold cross-validation combined with a grid parameter search where the best model of each algorithm was chosen to maximize the area under the ROC curve (AUC). Finally, each model was tested using the remaining 30% of the data.

We evaluated the performance of several tree-based approaches, including plain Decision Trees, Random Forests, Gradient-boosted trees (GBT)s, XGBoost and LigthGBM. They are known to produce acceptable results without requiring a lot of feature engineering. Moreover, since the implementations we use can already handle categorical features, no feature transformation was applied for those approaches. For plain Decision Trees and Random Forests the split was chosen to maximize the Information Gain (IG), according to the equation 1, where D represents the dataset, s a split, and N the size of the dataset. The Impurity in each node was calculated using the Gini impurity equation [13]. Random forests, composed of several independent decision trees built from a random subset of data and features, are a popular choice that typically helps reducing variance and

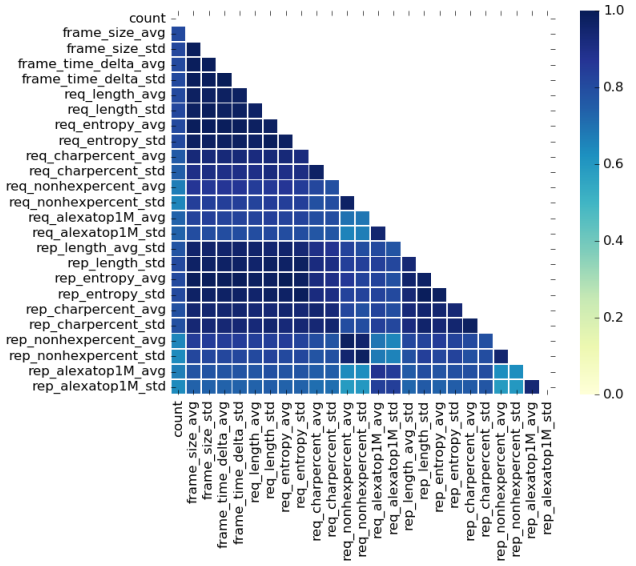

Figure 15: Feature Pairwise Mutual Information for DS3

over-fitting.

$$IG(D,s) = Impurity(D) - \frac{N_{left}}{N} Impurity(D_{left}) - \frac{N_{right}}{N} Impurity(D_{right}) \quad (1)$$

GBTs, on the other hand, use a sequential process where at each interaction they try to improve their result by incorporating the knowledge from the previous steps. We evaluated the native Spark implementation using a loss function according to the equation 2 [13]. Regarding XGBoost and LightGBM, despite none of them being natively supported by Spark, both offer an easy integration path, using Spark ML Dataframe pipelines. For XGBoost we used the `binary:logistic` objective option [19], whereas for LightGBM we used the `gbdt` boosting setting [49].

$$2\sum_{i=1}^{N} \log(1 + \exp(-2y_i F(x_i))) \quad (2)$$

We also evaluated the native Spark Multilayer Perceptron classifier (MLPC), based on feedforward artificial neural [13] network. We assessed different network architectures depending on the dataset. For DS1 we used 11, 7 and 2 as the number of nodes in the input, hidden and output layers, respectively. For DS2 we used 7, 5 and 2. Finally, for DS3, we used 25, 14 and 2. The number of nodes within the hidden layer was chosen based on the average of the number of nodes in the input and output layers. The hidden layer uses the sigmoid activation function, whereas the output layer uses the softmax function [13]. For the Naive Bayes case, we used a feature scaler to transform the features between 0 and 1.

Table 5
Summary of the key performance indicators for the DNS tunneling Detection using DS1

Technique	Accuracy	Precision	Recall	F1	AUC
Decision Tree	0.9914	0.9945	0.9872	0.9909	0.9912
Random Forests	0.9925	0.9982	0.9859	0.992	0.9921
GBT	0.9914	0.9945	0.9872	0.9909	0.9912
XGBoost	0.9989	1.0	0.9977	0.9989	0.9989
LightGBM	0.9989	0.9995	0.9982	0.9989	0.9989
Linear SVM	0.9776	0.9772	0.9754	0.9763	0.9775
MLPC	0.9895	0.9909	0.9868	0.9888	0.9893
NaiveBayes	0.9215	0.9115	0.9235	0.9174	0.9216
Logistic Regression	0.9202	0.9061	0.9271	0.9165	0.9206

Area under ROC (UAC)

4.3. Performance Indicators

This section provides an overview of the performance indicators of each method when applied to the different datasets, according to the following equations, where TP, TN, FP and FN stands for True Positives, True Negatives, False Positives and False Negatives respectively:

$$Precision = \frac{TP}{TP + FN} \quad (3)$$

$$Recall = \frac{TP}{TP + FP} \quad (4)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

$$F1 = \frac{2 * (Precision * Recall)}{Precision + Recall} \quad (6)$$

Table 5 shows the results of the various assessed machine learning algorithms for DS1. All the methods yield results above 90% of accuracy. Whereas no algorithm outperforms the remaining in all the measured indicators, Gradient Boosting trees, namely XGboost and LightGBM, provided the best results.

Table 6 shows the results of the various machine learning algorithms for DS2. As expected, the overall results were lower when compared to DS1. This can be explained due to the reduced number of features and records of DS2. This is a trade-off between a less accurate but faster and more generic model. Since we didn't use any Application layer-related features, this approach remains valid for detecting large deviations in the traffic patterns. Nevertheless, several methods surpass the 90% accuracy level. Tree-based methods obtained, once again, more consistent values with some advantage for the Random Forests approach, in this experiment. The large amounts of subsequent packets, when compared with a low traffic scenario as this case, can be easily

Table 6
Summary of the key performance indicators for the DNS tunneling Detection using DS2

Technique	Accuracy	Precision	Recall	F1	AUC
Decision Tree	0.8955	0.8776	0.5811	0.6992	0.7798
Random Forests	0.9237	0.8615	0.7568	0.8058	0.8623
GBT	0.9068	0.7887	0.7568	0.7724	0.8516
XGBoost	0.9124	0.8525	0.7027	0.7704	0.8353
LightGBM	0.8983	0.8276	0.6486	0.7273	0.8065
Linear SVM	0.8927	0.95	0.5135	0.6667	0.7532
MLPC	0.8909	0.8909	0.6203	0.7313	0.7982
Naive Bayes	0.8079	1.0	0.0811	0.15	0.5405
Logistic Regression	0.8898	0.8302	0.5946	0.6929	0.7812

Area under ROC (UAC)

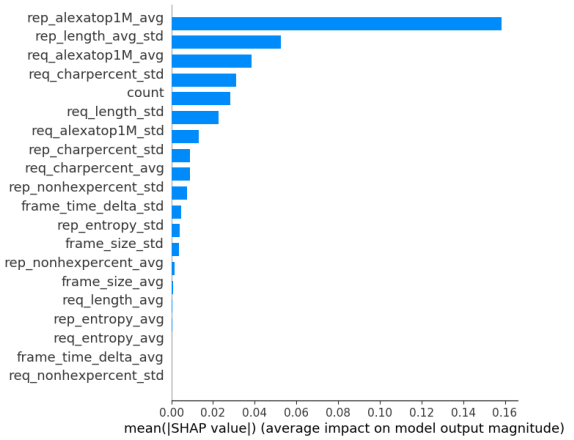

Figure 16: SHAP Summary plot for XGBoost Model using DS3 dataset

Table 7
Summary of the key performance indicators for the DNS tunneling Detection using DS3

Technique	Accuracy	Precision	Recall	F1	AUC
Decision Tree	0.9753	0.981	0.9699	0.9754	0.9753
Random Forests	0.9867	0.9887	0.985	0.9868	0.9867
GBT	0.981	0.9848	0.9774	0.9811	0.981
XGBoost	0.9886	0.9924	0.985	0.9887	0.9886
LightGBM	0.9734	0.9737	0.9737	0.9737	0.9734
Linear SVM	0.981	0.9923	0.9699	0.981	0.9811
MLPC	0.9867	0.9924	0.9812	0.9868	0.9868
Naive Bayes	0.9411	0.9916	0.891	0.9386	0.9416
Logistic Regression	0.981	0.9812	0.9812	0.9812	0.981

Area under ROC (UAC)

spotted. For larger networks, this might not be so accurate or might require additional fine-tuning of the time windows.

Table 7 shows the results of the various machine learning algorithms for DS3. By using a large number of features, including DNS-specific ones, there was an improvement when compared with DS2 results. As before, tree-based approaches produced the more interesting results, with a slight overall advantage for XGBoost with an AUC of 0.986. As with DS1, it is also remarkable that all the algorithms obtained an AUC score above 0.9.

Figure 16 shows the average impact of each feature on the XGboost model using the DS3 dataset. As expected, the presence of the DNS name in the Alexa top sites list was among the most significant features. DNS exfiltration attacks mostly use dedicated DNS records that are unlikely to appear in lists of popular domains. Similarly, the average length of DNS responses also played a significant role in the classification. This depends on the underlying network traffic, since high-throughput networks might benefit the attacker by smoothing such values.

The training times are another important indicator to understand how the different algorithms perform within the distributed computation platform. Such times are mainly influenced by: the number of records; the algorithm itself; its parameters; the capacity of distributing the workload and running tasks in parallel; the computational resource availability (e.g. CPU, GPU, FPGA); and the Spark cluster deployment setup (e.g. number of Spark workers, number of executors, number of cores per executor).

Table 8 represents the wall-clock time spent to train 70% of the original dataset (the training data) without any grid parameter search. These experiments were conducted using a 3-worker Spark cluster, each one running on a different virtual machine with 8 cores and 16GB RAM each, on a Dell PowerEdge R440 host with an Intel(R) Xeon(R) Gold 5120 CPU (28 vCPUs), 256GB RAM and 3.2TB datastore (4x 10K RPM SAS HDD in RAID6 via PERC H740P controller), running an ESXi 6.7 hypervisor instance. For experimental purposes, the Spark deployment ran on the top of a Docker container based on official NVIDIA/CUDA images with GPU passthrough. This allowed keeping the results consistent across all the tests, including a GPU-based approach comparison.

Given its greater number of records, overall training times for DS1 were slower than for DS2 and DS3. Such differences are expected to increase even more with larger datasets and might become a decision factor when choosing the right algorithm. For the same number of records, DS2 was, in general, faster than DS3, since we used a significantly less number of features (25 for DS3 versus 7 for DS2). Naive Bayes and Logistic regression were among the fastest methods, but also the most inaccurate. Linear SVM was significantly slower without any meaningful advantage when compared to tree-based approaches. On the other hand, LightGBM obtained a good balance between the training times and the actual classification results.

In an additional experiment, we compared the impact of available computational resources on the training time for larger datasets (using synthetic data generated from DS1). Figure 17 shows the XGBoost training times against differ-

Table 8
Summary of the training times(s)

Technique	DS1	DS2	DS3
Decision Tree	25.03	10.97	12.15
Random Forests	25.86	11.04	11.68
GBT	78.00	54.32	61.34
XGBoost	29.26	18.26	17.68
LightGBM	18.81	6.94	7.83
Linear SVM	173.84	56.18	52.60
MLPC	52.33	31.30	41.22
Naive Bayes	6.56	2.62	2.92
Logistic Regression	9.73	17.74	19.09

ent numbers of records. We compared the usage of 2 CPUs (Intel Core i7-6700HQ and Intel Xeon Gold 5120) using the `hist` XGBoost tree method and a NVIDIA GeForce GTX 1060 using `gpu_hist`.

For the Intel Core i7-6700HQ and NVIDIA GeForce GTX 1060 scenarios, we performed the tests using a single worker and a single executor with 7 cores (plus one additional core for the driver). For the Intel Xeon Gold 5120 CPU, we evaluated the single worker scenario with one executor and the cluster scenario with 3 workers (one executor per worker, with a total of 23 cores plus one additional core for the driver).

When handling inputs with fewer records, the Intel Xeon Gold 5120 CPU using a single worker provided the best results, since it avoided additional synchronization overhead. As expected, as the number of records increases, there is a clear advantage of pursuing a distributed approach with multiple workers or using a GPU-based approach. In our setup, after 5 million records, the NVIDIA GeForce GTX 1060 outperformed all the remaining approaches.

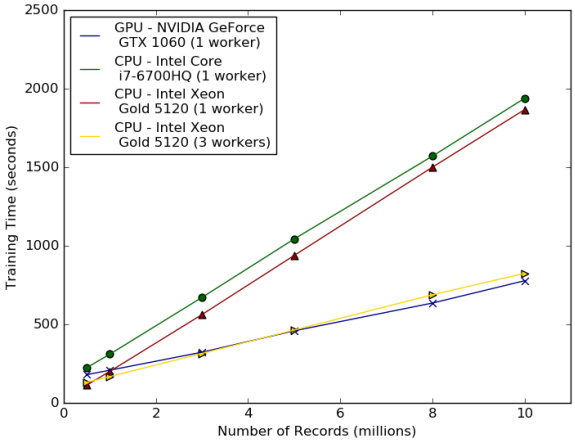

Figure 17: Comparison between CPU and GPU for XGBoost training times using synthetic data derived from DS1

4.4. Additional scenarios and SCADA support

The proposed framework is not limited to a single scenario or protocol. It was designed as a generic data-driven architecture, suited to cope with the majority of the surveyed types of attacks, which mainly refer to either generic layer 2/3 attacks that are also valid on SCADA networks (such as network scans and ARP-based attacks) or layer 7 attacks against specific SCADA protocols (e.g. [78]).

Some of those attacks might require complementing data sources and detection approaches, such as SCADA protocol-specific analysis or the usage of NIDS for detecting specific attack signatures. Nevertheless, the proposed framework is flexible enough to encompass, integrate and normalize those additional probes and data sources. Not all the attacks absolutely require complex machine-learning approaches. Some of them can also be detected using smart-probes and signature-based approaches. Nevertheless, by following a holistic approach, integrating different detection sources, the IADS becomes much more powerful and able to adjust to different scenarios.

The analysis of protocol-specific attacks or specialized probes is outside the scope of this paper. Nevertheless, the interested reader might refer to some of our previous works and open-source contributions (e.g. [78], [76], [77]) addressing layer 2/3 and layer 7 SCADA attacks.

4.5. Evaluation of Event Messaging

As mentioned before, the event messaging layer plays an important role in how the components communicate with each other. Therefore, it is important to understand how its settings can be used to tune the platform to meet the requirements for the next-generation of IACS, such as event processing capacity and scalability.

For the Kafka experiments, we set up a Kafka cluster with 3 brokers, each one running on a different virtual machine with 8 vCPUs, 32GB RAM and 850GB of disk. Similarly to the anomaly detection scenario, all the tests were conducted in a Dell PowerEdge R440 host with an Intel(R) Xeon(R) Gold 5120 CPU (28 vCPUs), 256GB RAM and 3.2TB datastore (4x 10K RPM SAS HDD in RAID6 via PERC H740P controller) running an ESXi 6.7 hypervisor instance. An additional virtual machine with 8 vCPUs, 8GB RAM and 20GB was also configured to work as a Kafka client (both producer and consumer). All the virtual machines, attached to the same vSwitch (with traffic shaping disabled), run CentOS 7.3 x64 with XFS. The broker instances were based on Confluent Docker images 4.0.0 (Apache Kafka 1.0.0).

Apache Kafka can be configured and optimized for different service goals, such as throughput, latency, durability or availability [18]. Within a SCADA environment, we expect to handle at least two different types of messages: (1) high-priority alarms, as a result of the output of specialized detection probes; and (2) low-priority events such as telemetry data. Such distinction is important to fine-tune the platform for each use case. In the first case, it is critical to ensure low latency, meaning that the event should be forwarded and processed as soon as it is received. In the second scenario, we could maximize the throughout by optimizing the number of events and the network overhead.

Based on a previous literature review [54], we evaluated the most significant configurations, including the message size, the acknowledgement level, the buffer size and the replication and partition number. For instance, a Kafka producer might be configured with the *batch-size* and *linger.ms* options to control whether the messages are sent as soon as they are ready or to add a delay based on size or time respectively.

Additionally, the producers can also send the events using different levels of acknowledgement, thus impacting the performance: asynchronous (i.e. without waiting for broker acknowledgements), synchronous with asynchronous replication (i.e. waits for an acknowledgement after the leader commit), or synchronous with synchronous replication (meaning the producer waits for the acknowledgement after a configurable number of replicas acknowledgements.

On the broker side, different settings impact the overall performance, both at the cluster and topic levels. Different topics can have different levels of replication and partitioning. The replication factor increases availability, which is critical for scenarios with strict fault-tolerance requirements (e.g. a topic with a replication number of 3 tolerates 2 broker failures). The partition number is used as a parallel approach to improve both broker and client performance. Different partitions from the same topic can spread among different brokers. In the same way, each partition can be assigned to different consumers. Nevertheless, a high number of partitions might increase downtime in the case of a broker failure, due to the partition reassignment process. Regarding message size, no limit is enforced by our IADS message data model. Based on our experiments, a typical message containing all the mandatory fields varies between 2KB and 5KB before Avro encoding, and between 500 and 1000 bytes after encoding.

This first test we conducted consisted of sending 1 million raw messages using the native Kafka client tool *kafka-producer-perf-test* with different configurations for the client and broker. The following results summarize the obtained values in terms of the number of messages per second, throughput and latency. All the tests were repeated 10 times with the confidence interval being computed using a Student's T distribution with a confidence level of 95 %.

Figure 18 shows how the message size negatively impacts the processed messages rate for different acknowledgement levels. The rate sharply dropped from over 100,000 messages per second (messages with 1 KByte) to 20,000 messages per second (7 KByte). The measured difference between full acknowledgement (acks = all) and no acknowledgement (acks = 0) decreased as the message size grows.

Figure 19 shows how message size affects performance. For 1KB messages, we achieved over 80,000 records per second, a value that decreased sharply for larger sizes. A similar trend is observed for throughput.

Figure 20 shows the effect of different topic configurations, namely the number of partitions and the replication factor. For fully synchronous producers the replication factor imposes a severe negative impact, as each message needs

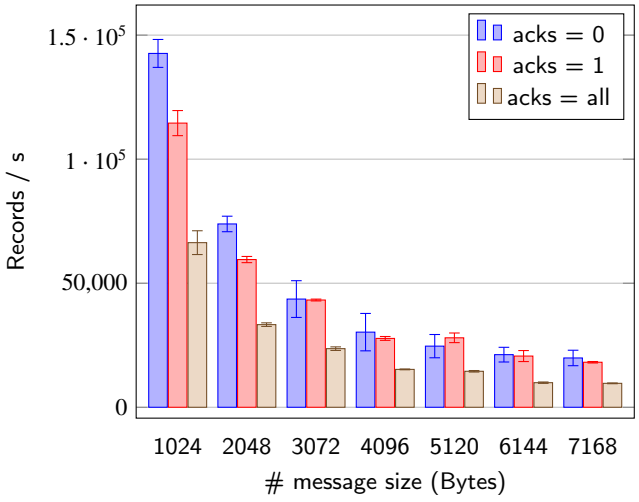

Figure 18: Average records / s versus the message size for a 1 Million messages production test. Three brokers, 3 partitions, replication factor 3. Error bars shows the 95% CI, n=10

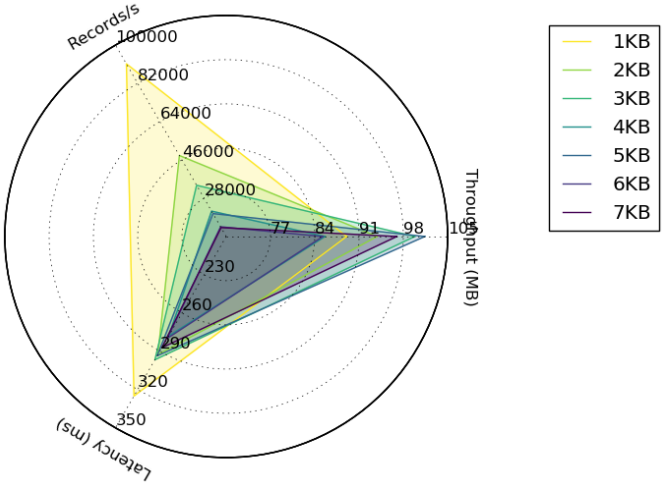

Figure 19: Batch Size 16384 bytes, Replication Factor 1, Number of partitions 1

to be committed by all the replicas before being acknowledged to the producer.

4.6. Scalability

As stressed before, next-generation IACS are expected to create new monitoring challenges, such as the need to design scalable monitoring solutions able to flexibly accommodate both small to large domains.

As already mentioned, our framework intrinsically achieves this. The key components it is built upon, Apache Kafka and Spark, were designed not only as high-performance systems but also as tools able to meet such heterogeneous use cases. The scalability of our framework in the scope of these two components has already been evaluated in this paper (cf. Sections 4.5 and 4.3, respectively).

Regarding Kafka, an elastic number of brokers ensure we can horizontally distribute the load across multiple in-

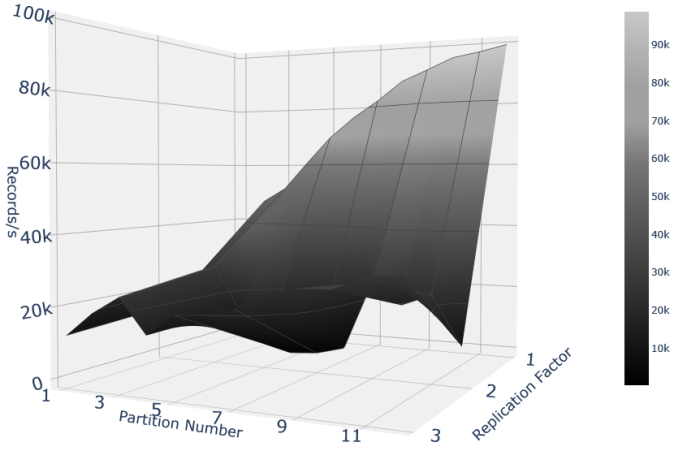

Figure 20: 3KB message size, Batch Size = 16384, Acknowledgments = all

stances, where each one might run on commodity hardware, up to high-end configurations, depending on the actual deployment scenario. Moreover, whereas there are no strict hardware minimum requirements [14] [17] [22], decent performance can be achieved starting from 32GB of RAM for small scenarios. CPU performance is not typically a major concern, but more powerful CPUs can be leveraged for encrypted communications, likely to be required in IADS deployments. Additionally, more cores can also unlock new levels of parallelization, and multiple disks / RAID setups can significantly improve I/O performance (note that the messages are always persisted to disk and optionally replicated).

Likewise, Spark is naturally designed to scale, both vertically and horizontally, with its highly optimized distributed and parallel approach – from small deployments, of 8-16 cores per machine, 8GB RAM and 4-8 disks per node, up to tenths of CPU cores and GBs of RAM per machine and network links over 10Gbps (since the data is maintained in memory, fast links helps to avoid network bottlenecks) [12]. Additionally, as described before, not only the number of nodes and executors impacts the cluster performance, as different resources types (such as GPUs) may also significantly boost the overall performance.

4.7. Wrap-up

We showcased the entire process of binary classification to detect DNS ex-filtration scenarios based on a wide range of supervised machine learning algorithms, demonstrating the IADS flexibility when incorporating and combining different algorithms.

Despite being constrained by commodity hardware, the evaluation of the proposed IADS shows promising results. Apache Spark has proven to be a good match for the implementation of the SIEM component, providing efficient distributed computation capabilities, as well as a unified approach for both streaming processing and batch processing. The SIEM component is a key component to address complex anomalies that cannot be easily detected by signature or threshold-based mechanisms.

For the Kafka-based event transport and preprocessing mechanisms, results have proven its ability to deal with different scenario requirements such as low-latency, high-throughput or durability, while keeping adequate performance levels.

5. Conclusions

Over recent years, most IACS-related security developments have been focused on SCADA protocols and their vulnerabilities, an approach that has spawned a multitude of SCADA security-related components and solutions. Apart from this, the majority of the research and industry efforts seems to have been narrowed to theoretical or specific problems.

Taking a somehow different path, the framework hereby presented enables the integration of different techniques and algorithms, rather than being a replacement for existing domain-specific detection tools. Common signature-based probes and dedicated smart probes are still useful for detecting local anomalies and known vulnerabilities. Nevertheless, new paradigms such as IIoT or 5G are pushing the focus to integration, performance and scalability. The proposed IADS is therefore a step beyond, constituting a framework where multiple heterogeneous components can be integrated and used together, providing an efficient and holistic approach for monitoring the security of a complex SCADA environment. The intermediate processing layer and the usage of domain processors represent an important step towards supporting edge computing for optimization, preprocessing and filtering of what is pushed to the upper layers.

On the other hand, the usage of such highly flexible and distributed approaches brings new challenges, which still need further research.

The most obvious challenge refers to the increased complexity of managing the platform itself. Ideally, the framework should also integrate zero-touch solutions, including self-monitoring and self-healing capabilities which would require minimal human-intervention.

Likewise, the ML process itself should not represent a complex task for the operator. It should be automated as much as possible, in line with the emerging MLOp concept [61]. We have presented a supervised anomaly detection use case (based on the DNS exfiltration scenario) which still required previous training and labeling. The feasibility and practical evaluation of the numerous anomaly detection algorithms proposed in the literature for other attack scenarios also needs further work.

Finally, in future, the integration of advanced privacy-preserving mechanisms to ensure data privacy (e.g. in the case of external data computation with a ML algorithm running on a third-party server in the cloud) should also be addressed.

6. Acknowledgments

This work was supported by the ATENA H2020 EU Project (H2020-DS-2015-1 Project 700581).

References

[1] Confluent, Inc, . Confluent schema management. URL: https://docs.confluent.io/current/schema-registry/index.html.

[2] International Electrotechnical Commission, a. Communication networks and systems for power utility automation - all parts. URL: https://webstore.iec.ch/publication/6028.

[3] International Electrotechnical Commission, b. Industrial communication networks - network and system security - part 1-1: Terminology, concepts and models. URL: https://www.isa.org/store/products/product-detail/?productId=116720.

[4] Aceto, G., Persico, V., Pescapé, A., 2019. A survey on information and communication technologies for industry 4.0: state of the art, taxonomies, perspectives, and challenges. IEEE Communications Surveys & Tutorials .

[5] Agrawal, S., Agrawal, J., 2015. Survey on anomaly detection using data mining techniques. Procedia Computer Science 60, 708–713. URL: http://dx.doi.org/10.1016/j.procs.2015.08.220, doi:10.1016/j.procs.2015.08.220.

[6] Alexa Internet, Inc, 2020. Alexa - top sites. URL: https://www.alexa.com/topsites.

[7] Andrew Banks, Ed Briggs, K.B., Gupta, R., 2019. Mqtt 5.0 - oasis standard. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html.

[8] Anton, S.D.D., Sinha, S., Schotten, H.D., 2019. Anomaly-based Intrusion Detection in Industrial Data with SVM and Random Forests URL: http://arxiv.org/abs/1907.10374, arXiv:1907.10374.

[9] Apache Foundation, . Welcome to apache avro! URL: https://avro.apache.org/.

[10] Apache Software Foundation, a. Apache kafka. URL: https://kafka.apache.org/.

[11] Apache Software Foundation, b. Apache spark™ - unified analytics engine for big data. URL: https://spark.apache.org/.

[12] Apache Software Foundation, 2019a. Apache spark: Hardware provisioning. URL: https://spark.apache.org/docs/2.4.4/hardware-provisioning.html.

[13] Apache Software Foundation, 2019b. Machine learning library (mllib) guide. URL: https://spark.apache.org/docs/2.4.4/ml-guide.html.

[14] Apache Software Foundation, 2020. Apache kafka documentation. URL: https://kafka.apache.org/documentation/.

[15] Basumallik, S., Ma, R., Eftekharnejad, S., 2019. Packet-data anomaly detection in PMU-based state estimator using convolutional neural network. International Journal of Electrical Power and Energy Systems 107, 690–702. URL: https://doi.org/10.1016/j.ijepes.2018.11.013, doi:10.1016/j.ijepes.2018.11.013.

[16] Bowes, R., 2019. dnscat2. URL: https://github.com/iagox86/dnscat2.

[17] Bromhead, B., Penchikala, S., . Apache kafka: Ten best practices to optimize your deployment. URL: https://www.infoq.com/articles/apache-kafka-best-practices-to-optimize-your-deployment/.

[18] Byzek, Y., 2019. Optimizing your apache kafka deployment. URL: https://cdn.confluent.io/wp-content/uploads/Optimizing_Your_Apache_Kafka_Deployment_White_Paper.pdf.

[19] Chen, T., Guestrin, C., 2016. XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA. pp. 785–794. URL: http://doi.acm.org/10.1145/2939672.2939785, doi:10.1145/2939672.2939785.

[20] Chen, W., Tianjiao, L., Yu, T., Dongsheng, X., 2019. Multi-level adaptive coupled method for industrial control networks safety based on machine learning 120, 268–275. doi:10.1016/j.ssci.2019.07.012.

[21] Cherepanov, A., 2017. Win32/industroyer, a new threat for industrial control systems. White paper, ESET (June 2017) .

[22] Confluent, Inc, 2019. Confluent documentation. URL: https://docs.confluent.io/home/overview.html.

[23] Cruz, T., Barrigas, J., Proença, J., Graziano, A., Panzieri, S., Lev, L., Simões, P., 2015. Improving network security monitoring for industrial control systems, in: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), IEEE. pp. 878–881.

[24] Demertzis, K., Iliadis, L., Kikiras, P., 2019. Cyber-Typhon : An Online Multi-task Anomaly Detection Framework .

[25] Dewa, Z., Maglaras, L.A., 2016. Data Mining and Intrusion Detection Systems 7, 62–71.

[26] Ding, D., Han, Q.l., Xiang, Y., Ge, X., Zhang, X.m., 2018. Neurocomputing A survey on security control and attack detection for industrial cyber-physical systems. Neurocomputing 275, 1674–1683. URL: https://doi.org/10.1016/j.neucom.2017.10.009, doi:10.1016/j.neucom.2017.10.009.

[27] Diogo, J., . Pccc dos packet rule. URL: https://marc.info/?l=snort-sigs&m=155024879909013.

[28] DNP Users Group, 2005. A dnp3 protocol primer. URL: https://www.dnp.org/Portals/0/AboutUs/DNP3%20Primer%20Rev%20A.pdf.

[29] D'Angelo, G., Ficco, M., Palmieri, F., 2020. Malware detection in mobile environments based on autoencoders and api-images. Journal of Parallel and Distributed Computing 137, 26–33.

[30] Ekman, E., Andersson, B., 2019. iodine. URL: https://github.com/yarrick/iodine.

[31] Elasticsearch B.V., . Elastic stack: Elasticsearch, kibana, beats & logstash | elastic. URL: https://www.elastic.co/elastic-stack.

[32] European Commission, 2014. Benchmarking smart metering deployment in the eu-27 with a focus on electricity. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52014SC0188.

[33] Feinstein, B., Curry, D., Debar, H., 2007. The Intrusion Detection Message Exchange Format (IDMEF). RFC 4765. URL: https://rfc-editor.org/rfc/rfc4765.txt, doi:10.17487/RFC4765.

[34] Foglietta, C., Masucci, D., Palazzo, C., Santini, R., Panzieri, S., Rosa, L., Cruz, T., Lev, L., 2018. From detecting cyber-attacks to mitigating risk within a hybrid environment. IEEE Systems Journal 13, 424–435.

[35] de Freitas, M.B., Quitério, P., Rosa, L., Cruz, T., Simões, P., . Sdn-assisted containerized security and monitoring components.

[36] de Freitas, M.B., Rosa, L., Cruz, T., Simões, P., 2018. Sdn-enabled virtual data diode, in: Computer Security. Springer, pp. 102–118.

[37] Gao, J., Dong, X., Lu, T., 2019. Omni SCADA Intrusion Detection Using Deep Learning Algorithms doi:10.13140/RG.2.2.33417.80488.

[38] Gao, Y., Xie, X., Parekh, M., Bajramovic, E., 2016. SIEM: Policy-based monitoring of SCADA systems, in: Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI), pp. 559–570.

[39] Ghaeini, H.R., Tippenhauer, N.O., 2016. HAMIDS: Hierarchical monitoring intrusion detection system for industrial control systems, in: CPS-SPC 2016 - Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and PrivaCy, co-located with CCS 2016, pp. 103–111. doi:10.1145/2994487.2994492.

[40] Ghosh, S., Sampalli, S., 2019. A Survey of Security in SCADA Networks: Current Issues and Future Challenges. IEEE Access PP, 1–1. doi:10.1109/access.2019.2926441.

[41] Goh, J., Adepu, S., Junejo, K.N., Mathur, A., 2016. A dataset to support research in the design of secure water treatment systems, in: International Conference on Critical Information Infrastructures Security, Springer. pp. 88–99.

[42] Handa, A., Sharma, A., Shukla, S.K., 2019. Machine learning in cybersecurity : A review , 1–7doi:10.1002/widm.1306.

[43] Henriques, J., Caldeira, F., Cruz, T., Simões, P., 2018. On the use of ontology data for protecting critical infrastructures. Journal of Information Warfare 17, 38–55.

[44] Henriques, J., Caldeira, F., Cruz, T., Simões, P., 2020. Combining k-means and xgboost models foranomaly detection using log dataset. Electronics (under revision) .

[45] Iturbe, M., Garitano, I., Zurutuza, U., Uribeetxeberria, R., 2017. Towards Large-Scale, Heterogeneous Anomaly Detection Systems in Industrial Networks: A Survey of Current Trends. Security and Communication Networks 2017. doi:10.1155/2017/9150965.

[46] Jiang, J., Zhao, X., Wallace, S., Cotilla-Sanchez, E., Bass, R., 2017. Mining PMU Data Streams to Improve Electric Power System Resilience , 95–102doi:10.1145/3148055.3148082.

[47] Kang, B., McLaughlin, K., Sezer, S., 2016. Towards A Stateful Anal-

ysis Framework for Smart Grid Network Intrusion Detection , 124–131doi:10.14236/ewic/ics2016.14.

[48] Kasperky, 2019. Threat landscape for industrial automation systems h2-2019 URL: https://ics-cert.kaspersky.com/media/KASPERSKY_H22019_ICS_REPORT_FINAL_EN.pdf.

[49] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y., 2017. Lightgbm: A highly efficient gradient boosting decision tree, in: Advances in neural information processing systems, pp. 3146–3154.

[50] Keliris, A., Salehghaffari, H., Cairl, B., Krishnamurthy, P., Maniatakos, M., Khorrami, F., 2017. Machine learning-based defense against process-Aware attacks on Industrial Control Systems. Proceedings - International Test Conference , 1–10doi:10.1109/TEST.2016.7805855.

[51] Khan, I.A., Pi, D., Khan, Z.U., 2019. HML-IDS : A Hybrid-Multilevel Anomaly Prediction Approach for Intrusion Detection in SCADA Systems. IEEE Access 7, 89507–89521. doi:10.1109/ACCESS.2019.2925838.

[52] Khodabakhsh, A., Ari, I., Bakir, M., 2017. Cloud-based Fault Detection and Classification for Oil & Gas Industry i, 1–6. URL: http://arxiv.org/abs/1705.04583, arXiv:1705.04583.

[53] Kravchik, M., Shabtai, A., 2018. Detecting Cyber Attacks in Industrial Control Systems Using Convolutional Neural Networks. Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy - CPS-SPC '18 , 72–83URL: http://dl.acm.org/citation.cfm?doid=3264888.3264896, doi:10.1145/3264888.3264896, arXiv:arXiv:1806.08110v2.

[54] Kreps, J., 2014. Benchmarking apache kafka: 2 million writes per second (on three cheap machines). URL: https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines.

[55] Langner, R., 2013. To kill a centrifuge: A technical analysis of what stuxnet's creators tried to achieve. The Langner Group .

[56] Lee, J., Jeon, J., Lee, C., Lee, J., Cho, J., 2016. An implementation of log visualization system combined SCADA Honeypot , 441–444.

[57] Leszczyna, R., 2019. Cybersecurity in the electricity sector .

[58] Lin, H., Slagell, A., Di Martino, C., Kalbarczyk, Z., Iyer, R.K., 2013. Adapting bro into scada: building a specification-based intrusion detection system for the dnp3 protocol, in: Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop, pp. 1–4.

[59] McKinnon, C., Carroll, J., McDonald, A., Koukoura, S., Infield, D., Soraghan, C., 2020. Comparison of new anomaly detection technique for wind turbine condition monitoring using gearbox scada data. Energies 13, 5152.

[60] Menze, T., 2019. The state of industrial cybersecurity 2019 .

[61] Merritt, R., 2020. What is mlops? URL: https://blogs.nvidia.com/blog/2020/09/03/what-is-mlops/.

[62] Modbus Organization, Inc, 2006. Modbus application protocol specification v1.1b. URL: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf.

[63] Morgan Stanley Research, 2015. The internet of things and the new industrial revolution.

[64] Morris, T., Gao, W., . Industrial control system network traffic data sets to facilitate intrusion detection system research. Critical infrastructure protection VIII—8th IFIP WG 11, 17–19.

[65] Nadler, A., Aminov, A., Shabtai, A., 2019. Detection of malicious and low throughput data exfiltration over the dns protocol. Computers & Security 80, 36–53.

[66] Nazir, S., Patel, S., Patel, D., 2017. Assessing and augmenting SCADA cyber security: A survey of techniques. Computers and Security 70, 436–454. URL: http://dx.doi.org/10.1016/j.cose.2017.06.010, doi:10.1016/j.cose.2017.06.010.

[67] Nguyen, V.Q., Van Ma, L., Kim, J.Y., Kim, K., Kim, J., 2018. Applications of anomaly detection using deep learning on time series data. Proceedings - IEEE 16th International Conference on Dependable, Autonomic and Secure Computing, IEEE 16th International Conference on Pervasive Intelligence and Computing, IEEE 4th International Conference on Big Data Intelligence and Computing and IEEE 3 , 393–396doi:10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00078.

[68] Nisioti, A., Mylonas, A., Yoo, P.D., Katos, V., 2018. From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods. IEEE Communications Surveys and Tutorials 20, 3369–3388. doi:10.1109/COMST.2018.2854724.

[69] Nuojua, V., David, G., Hämäläinen, T., 2017. Dns tunneling detection techniques–classification, and theoretical comparison in case of a real apt campaign, in: Internet of Things, Smart Spaces, and Next Generation Networks and Systems. Springer, pp. 280–291.

[70] ODVA, . Cip specifications library. URL: https://www.odva.org/Technology-Standards/Common-Industrial-Protocol-CIP/CIP-Specifications-Library.

[71] OPC Foundation, 2020. Opc unified architecture (ua). URL: https://opcfoundation.org/about/opc-technologies/opc-ua/.

[72] Open Information Security Foundation, . Open source ids / ips / nsm engine. URL: https://github.com/OISF/suricata.

[73] Phillips, B., Gamess, E., Krishnaprasad, S., 2020. An evaluation of machine learning-based anomaly detection in a scada system using the modbus protocol, in: Proceedings of the 2020 ACM Southeast Conference, pp. 188–196.

[74] Ramotsoela, D.T., Hancke, G.P., Abu-Mahfouz, A.M., 2019. Attack detection in water distribution systems using machine learning. Human-centric Computing and Information Sciences 9. URL: https://doi.org/10.1186/s13673-019-0175-8, doi:10.1186/s13673-019-0175-8.

[75] Rosa, L., a. New snort rules for pcom protocol. URL: https://marc.info/?l=snort-sigs&m=154746968717558.

[76] Rosa, L., b. Scada cip enum scan. URL: https://github.com/nmap/nmap/pull/1539.

[77] Rosa, L., c. Scada scan to collect information from unitronics plcs via pcom protocol. URL: https://github.com/nmap/nmap/pull/1445.

[78] Rosa, L., Freitas, M., Mazo, S., Monteiro, E., Cruz, T., Simões, P., 2019. A comprehensive security analysis of a scada protocol: From osint to mitigation. IEEE Access 7, 42156–42168.

[79] Rosa, L., de Freitas, M.B., Henriques, J., Quitério, P., Caldeira, F., Cruz, T., Simões, P., 2020. Evolving the security paradigm for industrial iot environments, in: Cyber Security of Industrial Control Systems in the Future Internet Environment. IGI Global, pp. 69–90.

[80] Shitharth, S., Prince Winston, D., 2017. An enhanced optimization based algorithm for intrusion detection in SCADA network. Computers and Security 70, 16–26. URL: https://doi.org/10.1016/j.cose.2017.04.012, doi:10.1016/j.cose.2017.04.012.

[81] Sokolov, A.N., Alabugin, S.K., Pyatnitsky, I.A., 2019. Traffic modeling by recurrent neural networks for intrusion detection in industrial control systems, in: 2019 International Conference on Industrial Engineering, Applications and Manufacturing, ICIEAM 2019, IEEE. pp. 1–5. doi:10.1109/ICIEAM.2019.8742961.

[82] Sundararajan, A., Wei, L., Khan, T., Sarwat, A.I., Rodrigo, D., 2018. A Tri-Modular Framework to Minimize Smart Grid Cyber-Attack Cognitive Gap in Utility Control Centers. Proceedings - Resilience Week 2018, RWS 2018 , 117–123doi:10.1109/RWEEK.2018.8473503.

[83] Taormina, R., Galelli, S., Tippenhauer, N.O., Salomons, E., Ostfeld, A., Eliades, D.G., Aghashahi, M., Sundararajan, R., Pourahmadi, M., Banks, M.K., Brentan, B.M., Herrera, M., Rasekh, A., Campbell, E., Montalvo, I., Lima, G., Izquierdo, J., Haddad, K., Gatsis, N., Taha, A., Somasundaram, S.L., Ayala-Cabrera, D., Chandy, S.E., Campbell, B., Biswas, P., Lo, C.S., Manzi, D., Luvizotto, Jr, E., Barker, Z.A., Giacomoni, M., Pasha, M.F.K., Shafiee, M.E., Abokifa, A.A., Housh, M., Kc, B., Ohar, Z., 2018. The battle of the attack detection algorithms: Disclosing cyber attacks on water distribution networks. Journal of Water Resources Planning and Management 144, 04018048. doi:10.1061/(ASCE)WR.1943-5452.0000969.

[84] Terzi, D.S., Terzi, R., Sagiroglu, S., 2017. Big data analytics for network anomaly detection from netflow data, in: 2017 International Conference on Computer Science and Engineering (UBMK), IEEE. pp. 592–597.

[85] Udd, R., Asplund, M., Nadjm-Tehrani, S., Kazemtabrizi, M., Ekstedt,

M., 2016. Exploiting bro for intrusion detection in a scada system, in: Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security, pp. 44–51.

[86] Vávra, J., Hromada, M., 2016. Comparison of the intrusion detection system rules in relation with the scada systems, in: Computer Science On-line Conference, Springer. pp. 159–169.

[87] Wong, K., Dillabaugh, C., Seddigh, N., Nandy, B., 2017. Enhancing suricata intrusion detection system for cyber security in scada networks, in: 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), IEEE. pp. 1–5.

[88] Yang, H., Cheng, L., Chuah, M.C., 2019. Deep-Learning-Based Network Intrusion Detection for SCADA Systems. 2019 IEEE Conference on Communications and Network Security (CNS) , 1–7doi:10.1109/cns.2019.8802785.

[89] YANG, W., Haider, S.N., ZOU, J.h., ZHAO, Q.c., 2017. Industrial Big Data Platform Based on Open Source Software, pp. 649–658. doi:10.2991/cnct-16.2017.90.

[90] Yassine, S., Khalife, J., Chamoun, M., El Ghor, H., 2018. A survey of dns tunnelling detection techniques using machine learning., in: BDCSIntell, pp. 63–66.

[91] Yu, B., Smith, L., Threefoot, M., Olumofin, F.G., 2016. Behavior analysis based dns tunneling detection and classification with big data technologies., in: IoTBD, pp. 284–290.

Luis Rosa received the M.Sc. degree in informatics engineering from the Higher School of Technology and Management, Polytechnic Institute of Coimbra, Coimbra, Portugal, in 2013. He is currently working toward the Ph.D. degree in informatics engineering at the University of Coimbra. He is a Junior Researcher at the Centre for Informatics and Systems, University of Coimbra, where he participates in several research projects in those fields. His research interests include security, event management, and critical infrastructure protection.

Tiago Cruz received the Ph.D. degree in informatics engineering from the University of Coimbra, Coimbra, Portugal, in 2012. He has been an Assistant Professor in the Department of Informatics Engineering, University of Coimbra, since December 2013. His research interests include areas such as management systems for communications infrastructures and services, critical infrastructure security, broadband access network device and service management, Internet of Things, software-defined networking, and network function virtualization (among others). He is the author of more than 40 publications, including chapters in books, journal articles, and conference papers. Dr. Cruz is a member of the IEEE Communications Society.

Miguel Borges de Freitas received the M.Sc. degree in Chemical Engineering from the Technical University of Lisbon in 2016 and the M.Sc. degree in Informatics Engineering from the University of Coimbra in 2018. He is currently a Junior Researcher at the Center for Informatics and Systems, University of Coimbra. His research interests target Software Defined Networking, Network Function Virtualization, cybersecurity and critical infrastructure protection. He is also a collaborator for some open-source projects.

Pedro Quitério received the M.Sc. degree in Informatics Engineering from the University of Coimbra, in 2018, where he is currently a Researcher with the Center for Informatics and Systems. His research interests include web development, data visualization, event management and critical infrastructure protection.

João Henriques is a PhD candidate in Science and Information Technology at the University of Coimbra (UC) and Assistant Professor at the Department of Informatics Engineering at the Polytechnic Institute of Viseu (IPV). His research interests at the Center for Informatics and Systems (CISUC) at UC includes forensic and audit compliance for critical infrastructures protection. He also remains as Software Engineer in the private sector.

Filipe Caldeira is an Adjunct Professor at the Polytechnic Institute of Viseu, Portugal. He is a researcher at the CISeD research centre of the Polytechnic Institute of Viseu and at the Centre for Informatics and Systems of the University of Coimbra. His main research interests include ICT security, namely, trust and reputation systems, Smart Cities and Critical Infrastructure Protection. His research papers were published in various international conferences, journals and book chapters. He has been recently involved in some international and national research projects.

Edmundo Monteiro is currently a Full Professor with the University of Coimbra, Portugal. He has more than 30 years of research experience in the field of computer communications, wireless networks, quality of service and experience, network and service management, and computer and network security. He participated in many Portuguese, European, and international research projects and initiatives. His publication list includes over 200 publications in journals, books, and international refereed conferences. He has co-authored nine international patents. He is a member of the Editorial Board of the Springer Wireless Networks Journal and is involved in the organization of many national and international conferences and workshops. He is a Senior Member of the IEEE Communications Society and the ACM Special Interest Group on Communications. He is also a Portuguese Representative in IFIP TC6 (Communication Systems).

Paulo Simões received the Doctoral degree in informatics engineering from the University of Coimbra, Coimbra, Portugal, in 2002. He is an Associate Professor in the Department of Informatics Engineering, University of Coimbra, where he regularly leads technology transfer projects for industry partners such as telecommunications operators and energy utilities. His research interests include network and infrastructure management, security, critical infrastructure protection, and virtualization of networking and computing resources. He has more than 150 publications in refereed journals and conferences. Dr. Simões is a senior member of the IEEE Communications Society.