international journal of critical infrastructure protection 42 (2023) 100613

Contents lists available at ScienceDirect

International Journal of Critical Infrastructure Protection

journal homepage: www.elsevier.com/locate/ijcip

A forensics and compliance auditing framework for critical infrastructure protection

João Henriques [a,b], Filipe Caldeira [b,c], Tiago Cruz [a,*], Paulo Simões [a]

[a] University of Coimbra, Centre for Informatics and Systems of the University of Coimbra, Department of Informatics Engineering, 3030-290 Coimbra, Portugal
[b] Informatics Department, Polytechnic of Viseu, 3504-510 Viseu, Portugal
[c] CISeD—Research Centre in Digital Services, Polytechnic of Viseu, 3504-510 Viseu, Portugal

ARTICLE INFO

Keywords:
Forensics
Compliance auditing
Critical infrastructure protection
Cybersecurity
Big data
Data analytics
Distributed computing

ABSTRACT

Contemporary societies are increasingly dependent on products and services provided by Critical Infrastructure (CI) such as power plants, energy distribution networks, transportation systems and manufacturing facilities. Due to their nature, size and complexity, such CIs are often supported by Industrial Automation and Control Systems (IACS), which are in charge of managing assets and controlling everyday operations.

As these IACS become larger and more complex, encompassing a growing number of processes and interconnected monitoring and actuating devices, the attack surface of the underlying CIs increases. This situation calls for new strategies to improve Critical Infrastructure Protection (CIP) frameworks, based on evolved approaches for data analytics, able to gather insights from the CI.

In this paper, we propose an Intrusion and Anomaly Detection System (IADS) framework that adopts forensics and compliance auditing capabilities at its core to improve CIP. Adopted forensics techniques help to address, for instance, post-incident analysis and investigation, while the support of continuous auditing processes simplifies compliance management and service quality assessment.

More specifically, after discussing the rationale for such a framework, this paper presents a formal description of the proposed components and functions and discusses how the framework can be implemented using a cloud-native approach, to address both functional and non-functional requirements. An experimental analysis of the framework scalability is also provided.

1. Introduction

Current living standards increasingly depend on the essential products and services provided by CIs such as electricity generation and distribution platforms, transportation systems and manufacturing facilities. Thus, disruptions or downtime in such environments can be disastrous, potentially leading to the service interruptions, causing significant monetary losses, physical destruction of valuable assets or even loss of human lives. For these, and other reasons, such infrastructures are becoming more and more attractive targets for cyber attacks [1].

While undoubtedly security is a serious concern, with potential ramifications in terms of safety and service availability, the ongoing evolution of the CI paradigm is not helping either. In fact, CIs are becoming more complex, as a result of the increasing number of interconnected devices, sensors and actuators, often distributed in the field, and the increasing amount and heterogeneity of information being exchanged in the network among system components. Smart Grids, oil and gas distribution, transportation systems and Industry 4.0, among others, are pushing the boundaries of the classic infrastructure model, moving towards a new generation of IACS. The gradual introduction of Industrial Internet of Things (IIoT) [2] further reinforces this trend.

With CIs becoming more complex and intertwined in their surrounding environment, the pressure to provide increased availability and reliability levels also grows, as a result of the higher degree of dependency and coupling between equipment, infrastructures and services. This paradigm shift calls for the development of proper security measures, able to identify and prevent the growing number of threats that could compromise those systems and avoid disasters, while also keeping up with the scale and complexity requirements of protected infrastructures.

While the importance of developing suitable CIP mechanisms is generally acknowledged by most ecosystem players, a great deal of effort has been focused on developing prevention, detection and mitigation tools and techniques, and not as much on aspects such as

* Corresponding author.
E-mail addresses: jpmh@dei.uc.pt (J. Henriques), caldeira@estgv.ipv.pt (F. Caldeira), tjcruz@dei.uc.pt (T. Cruz), psimoes@dei.uc.pt (P. Simões).

https://doi.org/10.1016/j.ijcip.2023.100613
Received 26 October 2022; Received in revised form 8 March 2023; Accepted 13 June 2023
Available online 17 June 2023

forensics support or compliance auditing. Still, the latter should not to be disregarded, due to several reasons:

- The best practices for incident handling always encompass a "lessons learned" stage, that is only possible by means of post-mortem trace analysis, allowing experts to reconstruct the trail conducting to the root cause and generate valid and useful evidence.
- There is an extensive body of knowledge supporting the definition of proper procedures and guidelines for ongoing secure and safe operation, management and maintenance of CI. However, the effectiveness of such policies is only as good as the existing capabilities to continuously monitor their correct enforcement and application, something that becomes increasingly difficult to be manually undertaken as the infrastructure size and complexity grows.

In this paper, we propose a so-called Forensics and Compliance Auditing (FCA) framework designed to provide explicit support for these activities in the specific scope of CIP. It constitutes a domain-specific augmented flight recorder that, besides persisting relevant information, also incorporates compliance auditing capabilities addressing CI security policies and forensics capabilities for post-mortem event analysis, with the ability to scale by making use of a cloud-native approach. The architecture of the proposed framework enables an unified security solution in the context of distributed IACS, with tools that are easy to deploy, use, modify and extend. This approach allows to collect forensic evidence and ensures that future unforeseen incidents are avoided or identified to leverage classification capabilities throughout the assessment of deviations to reference behaviors. This will contribute to prevent risks from operational errors and cyber-attacks. Compliance auditing on the CI security policies also contributes to eventually minimize future security incidents. This allows to leverage the outcomes of forensic research in the benefit of auditing compliance, in which forensic tools can be reused to assess regulatory compliance.

The rest of this paper is organized as follows. Section 2 introduces the most relevant background and related work on the subject. Section 3 describes the proposed Forensic and Compliance Auditing framework. Section 4 presents the implemented proof-of-concept. Section 5 presents an experimental performance evaluation. Finally, Section 6 concludes the paper and discusses future work.

2. Background and related work

In this section we start by discussing related works on the topics of forensics (Section 2.1) and compliance auditing (Section 2.2). In Section 2.3 we introduce the reader to the concept of Security Information and Event Management (SIEM). Finally, we identify and discuss the key challenges of building FCA capabilities (Section 2.4).

2.1. Forensics

By definition, forensic science encompasses a set of activities and techniques for collection, examination and analysis of facts, records and all relevant evidence that can be used to reconstruct past events/incidents and, eventually, be legally used for criminal prosecution [3]. More specifically, Rani and Geethakumari [4] define computer forensics as the science allowing to identify, extract, preserve and describe the digital evidence stored in digital devices and networks that can be legally admissible in court for any cyber-crime or fraudulent act (for instance, a security breach or a directed attack). NIST [5] defines digital forensics as a scientific method for identifying, collecting, examining and analyzing data while preserving the integrity of the information and maintaining a strict chain of custody of data. In the course of a forensic investigation, it should be assured that all available digital evidence is not modified without appropriate authorization [6].

Forensic investigations aim to lookup for answers to the questions raised after an incident has occurred. According to Hunt [7], the main purpose of collecting forensically sound data is to seek for answers regarding *who* generated the incoming intrusion or outgoing data transfer, *what* kind of equipment and services were involved, and whether they *were* able to do this because of limitations of incoming or outgoing security mechanisms. In this scope, forensics data usually comes from two main sources: computer sources, such as files, logs and code, and network sources, such as observed network traffic.

The first guidelines on digital forensics were presented in the early 1990s by the International Association of Computer Investigative Specialists (IACIS). Later, the International Organization on Computer Evidence (IOCE) developed a set of principles for computer-based evidence, followed by similar guidelines. In general, all those guidelines recognize that digital evidence should be totally acquired and not be altered during and after the examination [8].

Regarding IACS and associated Supervisory Acquisition and Data Control (SCADA) systems forensics, the European Union Agency for Network and Information Security (ENISA) proposes five steps [9]: (1) examination of the system for the possible sources of evidence, (2) identification of the impact components, (3) collection of raw data (in a form that is unmodified), (4) analysis of evidence, and (5) documentation.

High-profile attacks against IACS such as Stuxnet [10], Dragonfly [11], Flame [12] and BlackEnergy [13,14] highlighted the relevance in taking into account forensic investigations. One of the main challenges in IACS forensics is to collect evidence without impacting the function of the system (e.g. creating delays in time-sensitive control loops).

Quick and Choo [15] highlighted the key challenges for digital forensic analysis, considering the ongoing growth in the volume of data seized and presented for analysis. The growth in data volumes and the increasing number of evidence items from a wide range of devices, such as multiple mobile devices and Internet of Things (IoT) devices, have compounded the challenges when conducting digital forensic investigations [16]. Koven et al. [17] also concluded that there is a lack of suitable tools for analyzing large datasets.

Javed et al. [18] provide a survey on digital forensics. Authors start by introducing computer forensic domains and forensic toolkits used for computer forensics. Then, they provide a comparative analysis based on the forensic tool characteristics. Finally, they summarize the current challenges and future research directions in computer forensics.

Casino et al. [19] also provide a review in the field of digital forensics. First, authors determined the main topics on digital forensics and identified their main challenges. Next, they highlighted procedural issues in terms of readiness, reporting and presentation, as well as ethics. Moreover, they also highlighted the European perspective in terms of privacy.

Rizvi et al. [20] surveyed the state-of-the-art in network forensics and the application of expert systems, machine learning, deep learning, and ensemble/hybrid approaches to a range of application areas in the field. These approaches included network traffic analysis, intrusion detection systems, IoT devices, cloud forensics, DNS tunneling, smart grid forensics, and vehicle forensics.

Ganesh et al. [21] provided a literature review on the application of forensics using Artificial Intelligence in the field of Cloud computing, IoT, and Blockchain technology.

Roussev and Richard [22] discussed the need for distributed forensic approaches, to leverage the performance benefits inherent to distributed computing. They proposed a digital forensic tool that centralizes data while distributing the computation over multiple devices, supporting preprocessing in the background and multiple concurrent searches.

Hunt and Slay [7] advocate the need for novel approaches regarding forensic analysis, with specialized engines supported by parallel

computing and flexible customized evidence analysis, even though they make no concrete proposals to address this need.

The investigation process pursues the reconstruction of events by trying to find out the connections between them, by rebuilding their timeline to be extracted as evidence. In this scope, Xie et al. [23] proposed to use provenance, the history or lineage of an object that explicitly represents the dependency relationship between the damaged files and the intrusion processes, rather than underlying system calls to detect and analyze intrusions. Valli [24] created a framework producing forensically verified signatures for Snort Intrusion Detection System (IDS) for known and published vulnerabilities of SCADA and control systems, thus enabling investigators to trace exploits during analysis.

To support the analysis of evidence, several schemas have been proposed to represent digital forensic information to be shared along the investigation chain [25–29], but these have not been widely adopted. DFAX, among others, tries to address the lack of consensus in the adoption of an ontology to support consistent representation and exchange of digital forensic information with an ontology that could be used for community consensus [30].

Another challenge in forensics is to grant data privacy protection during the digital forensic investigation, as discussed by Aminnezhad et al. [31]. Apart from containing potential evidence files, seized storage media may also contain owner's private data, such as private/-family pictures and videos, business related digital documents, medical diagnostic or treatment reports, commercial software with license information, and much more. Investigator's open access to these private files is a threat to the owner's data privacy [32].

Considering specific implementation proposals in the field of CI, Grammatikis et al. [33] presented an architecture aiming to protect Smart Grids (SG) by enhancing the situational awareness, detect timely cyber attacks and collecting appropriate forensic evidence. In that to include a forensics repository and a forensic readiness framework and mechanisms to support the reputation of each asset beyond the visualization capabilities of the cybersecurity incidents. The same team also suggested a multivariate Intrusion Detection System (IDS) by adopting both access control and outlier detection mechanisms [34]. They aimed to detect timely possible anomalies against IEC-104 as a set of standards that define systems used for remote control (SCADA) in electrical engineering and power system automation applications.

2.2. Compliance auditing

Compliance auditing is a systematic, independent, formal, structured and documented process applied proactively, aiming to verify the fulfillment of internal policies, external formal standards and/or legal requirements [35]. This process is typically performed by certified professionals, on behalf of specific stakeholders. Depending on the context, organizations may adopt rules and/or criteria defined either by themselves or externally imposed (as result of corporate laws, policies, procedures, customer requirements or external regulations).

Compliance auditing activities can help to identify and assess risks as the way to prevent threats. Moreover, they can contribute to catch violations by evaluating security properties (e.g. due to misconfigurations, exploits of vulnerabilities, or even the detection of attacks or intrusions).

In the specific scope of cybersecurity, for instance, many organizations have developed Information Security Management Systems (ISMS) in accordance with the International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 27001 [35] and ISO/IEC 27002 [36]. There are also initiatives for developing and implementing IACS-specific standards, such as the Special Publication 800–82 from the National Institute of Standards and Technology (NIST), [37] and ISO/IEC 27019 [38], the latter having evolved from ISO/IEC 27002 and being focused on the infosecurity operational aspects of the energy utility industry.

The Japanese Information Technology Promotion Agency (IPA) has also proposed the Embedded Device Security Assurance Certification Program with provisions for SCADA devices [39]. Non-security standards such as ISO 9001 and ISO/IEC 90003, focused on acquisition, supply, development, operation and maintenance of computer software and related support services, are also relevant since they somehow help regulate software supply-chain activities. Other cybersecurity standards benefiting from audit compliance support include, for instance, Payment Card Industry Data Security Standard (PCIDSS) [40] and NIST SP 800–53. IEC 62443-4-2 (Technical Security Requirements For IACS Components) [41] also defines a series of functional requirements encompassing logging/audit aspects related to storage, protection or accessibility.

There are a few research works addressing the verification of access control policies, typically resorting to formal reasoning mechanisms to verify application control expressed at the design time (for instance with eXtensible Access Control Markup Language, XACML) to dynamically enforce authorization by externalizing access controls. Fisler et al. [42] proposed Binary Decision Diagrams (BDD) and custom algorithms to check access-control policies. Ahn et al. [43] used answer set programming (ASP) and leverage existing ASP reasoning models to conduct policy verification. Arkoudas et al. [44] proposed a Satisfiability Modulo Theory policy analysis framework.

Other works explored the application of compliance auditing techniques to cloud security. Ullah et al. [45], for instance, proposed an architecture to build automated security compliance tools for cloud computing platforms, focusing on auditing remote administration and on diagnosing port protection and clock synchronization. Doelitzscher [46] implemented an on-demand audit architecture for Infrastructure as a Service (IaaS) clouds, based on software agents for identifying anomalies in IaaS clouds for auditing purposes. Microsoft also proposed SecGuru [47] to audit Azure datacenter network policies. Similarly, IBM introduced the QRadar [48] SIEM to collect and analyze events in the cloud, integrating several monitoring tools. Amazon also suggests using AWS CloudWatch and CloudTrail [49] for auditing purposes, as a web API offering logs and metrics data to their clients. Nevertheless, despite these efforts from the academia and the industry, there is still a lack of standards for compliance auditing techniques in Cloud domains [50].

Compliance auditing intimately relates to forensics processes because forensics tools and techniques can often be used as well for the assessment of compliance [51]. Data collected along forensics activities is often the best input for compliance auditing, since it offers detailed, time-stamped information about audited processes. By defining audit rules and automatically applying them to forensics data, it becomes possible to conduct much more detailed and embracing audits in order to detect potential non-conforming cases. By assuring the conformance with applicable laws, regulations, policies, and standards, auditing and compliance processes may contribute to increase the level of security of CI. One of the major challenges in compliance auditing is the existing gap between high-level guidelines and recommendations in the security standards and low-level logs provided by current systems.

2.3. Security information event management

In the scope of cybersecurity, SIEM systems are intimately related with FCA frameworks, since they are the main source of data stored and processed in the scope of FCA. Moreover, forensics analysis and compliance auditing are key security management activities.

SIEM systems collect and process security-related data from a wide variety of sources within organizations, including security controls, operating systems, network traffic and applications. Collected data is normalized by the SIEM systems and analyzed, for instance generating alerts when suspect activities are detected and providing activity reports on request. SIEM technology includes Security Information Management (SIM) and Security Event Management (SEM) [52].

SIM provides log management, analytics and compliance reporting, and SEM provides real-time monitoring and incident management for security-related events from networks, security devices, systems and applications. SIEM platforms typically support three primary use cases: advanced threat detection; basic security monitoring (log management, compliance reporting and basic real-time monitoring of selected security controls); and forensics and incident response.

González-Granadillo et al. [53] identified the usual components of a SIEM solution: Source Devices, Log Collection, Parsing and Normalization, Rule Engine and Correlation Engine, Log Storage and Monitoring. The authors also identified the key features to be considered within a SIEM: (1) correlation rules, (2) data sources, (3) real time processing, (4) data volume, (5) visualization, (6) data analytics, (7) performance, (8) forensics, (9) complexity, (10) scalability, (11) risk analysis, (12) storage, (13) price, (14) resilience, (15) reaction and reporting capabilities, (16) user and entity behavior analytics, and (17) security.

The increasing scale and complexity SIEM systems, as well as the increasing requirements of emerging security analytic technologies, have driven interest in alternative approaches for collecting and analyzing event data to identify advanced attacks. The combination of Elasticsearch, Logstash and Kibana from Elastic Stack, OpenSOC, Apache Metron and other tools, leveraged with (or natively using) Big Data platforms like Hadoop, offers data collection, management and analytics capabilities. Other security analytics platforms are available [54–59], and many open-source solutions have been developed supporting a wide spectrum of security-based analysis [60,61].

The already mentioned survey from González-Granadillo et al. [53] also highlighted the lack of digital forensics capabilities and limited network forensic capabilities in current SIEM, and found a generalized lack of compliance auditing capabilities. The same survey also points to the lack of capabilities to: (i) integrate new connectors or parsers to collect events or data, and (ii) to provide web Application programming interface (API)s or RESTful interfaces to collect them.

2.4. The challenge of building FCA capabilities

Despite the fact that Forensics and Compliance Auditing are different activities, both in terms of purpose and expected outcomes, there is a considerable amount of proximity between them, due to the fact that they often resort to the same data sources and similar information and context extraction techniques to gather and process evidence. Such similarities hint at the possibility of building both capabilities on top of a shared infrastructure, providing data acquisition, transport and processing pipelines, as well as persistence capabilities. This realization provides a strong rationale to support these processes by means of a converged approach, which is the main scope of the FCA framework concept outlined in this paper.

This section identifies and discusses the key challenges and requirements to be attained when building FCA capabilities, such as the framework we propose in this paper.

FCA frameworks need to deal with ongoing and past harmful events (e.g. attacks, threats, hardware malfunctioning), in order to support the identification and prosecution of attackers. That data will also support the identification of the root cause of incidents and, later on, the extraction of evidence for forensics purposes. This requires a deep analysis and correlation of the data emanated from various heterogeneous sources, such as the underlying CI (e.g. SCADA logs, sensor data) and the supporting Information and Communication Technologies (ICT) infrastructure (e.g. email and other service logs, network traffic and IDS systems).

The acquisition of data for forensics purposes will also help organizations to audit and enforce compliance with industry standards and regulatory requirements — allowing them to assess the correct application of standards, certification requirements, best practices, business rules or stakeholder security policies.

FCA processes become increasingly difficult to handle as the amount of acquired evidence grows. This calls for a balancing act between automation and interpretability, in order to remove the cognitive friction from the analyst and let him/her focus on evaluating the insights produced by the system. From this perspective, an FCA framework cannot restrict itself to providing acquisition and persistence mechanisms, under the risk of becoming increasingly useless as the amount and complexity of collected data grows, making any sort of manual analysis procedure difficult, slow and, ultimately, unfeasible.

This need, along with the rise in size and complexity of the modern distributed CIs under protection, poses increasing difficulties to the timely and effective collection and processing of security data. As a consequence, the security frameworks protecting those CIs also need to become more flexible and resilient, in order to quickly adapt to the changing conditions of the operational environment — including characteristics such as horizontal scaling, elasticity, adaptability, resiliency and fault-tolerance.

The concerns about cognitive friction also make it necessary to handle different granularity levels for the information to be provided to auditors and examiners. According to the forensics or compliance audit tasks at hand, in some cases only high-level information should be provided (in order to not overload the analyst) while, in other cases, it is necessary to drill-down to very detailed data (e.g. to identify the root cause of failures, in complex scenarios). This impacts storage and compute requirements and, indirectly, data retention strategies: while in some cases storage resources may be optimized at the cost of granularity (when later detailed analysis is not necessary), in other cases low-level data needs to be retained for long periods of time.

Further analysis into the nature of the FCA tasks and their applicability in different contexts is also instrumental in defining which are the most relevant data sources for activity and process profiling and/or monitoring. For instance, profiling users' behavior in forensics contexts often requires access to operational data from the ICT infrastructure, encompassing aspects such as: service usage historical data; network flows; system-level logs providing information about running processes; service events; file modifications; and any other important evidence, for that matter. Unsurprisingly, the same data sources can also be leveraged for compliance auditing purposes, to provide ongoing monitoring of conformity with adopted standards and guidelines.

The adoption of normalized common schemes is also important, to integrate the large number of heterogeneous data sources involved in FCA. Moreover, this normalization is not just a matter of adopting common formats, but also of properly handling non-synchronized data sources (e.g. clock skews) so data events may still be properly sequenced in the timeline.

3. Proposed FCA framework

In this section we present the proposed FCA framework. First, we describe the reference architecture (Section 3.1). Next, we specifically discuss each key component of the framework: Data Acquisition; Domain Processor; Cyber–physical IDS; Data Lake; CI Business Rules; Monitoring; Data Visualization; Analytics; Trust and Reputation Indicators; Orchestration; Actors and Roles.

3.1. Architecture

Fig. 1 shows the architecture of the proposed platform. In practice, this framework works like a domain-specific augmented flight recorder that complements typical security solutions by providing audit compliance and forensic tools. It supports forensics activities for identifying, extracting, preserving, and highlighting digital evidence. Regarding compliance auditing, it enables the assessment of compliance with standards, business rules, and policies.

The framework is able to dynamically scale horizontally, in order to adapt to different deployment scenarios and to different loads over

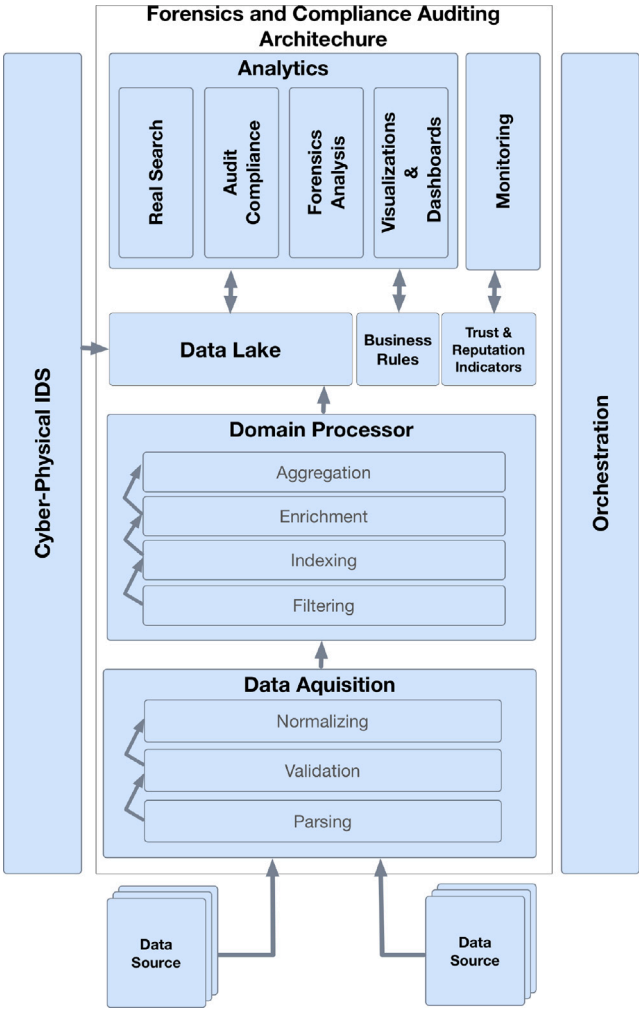

Fig. 1. Architecture of the proposed FCA framework.

time, while keeping the ability to collect, store and correlate large volumes of data from a large number of disparate and geographically dispersed sources.

The platform collects data from multiple sources, gathered into an unified view to enhance FCA capabilities such as digital evidence extraction. Data sources include, for instance, service and device logs, physical process logs, network traces, and Authentication Authorization and Accounting (AAA) sessions or physical access control systems, both for forensics and compliance auditing purposes. It is able to cope with high-throughput flows of structured and unstructured heterogeneous data, acquired from both internal (e.g. IACS) and external sources (e.g. corporate ICT systems) — therefore enabling a broader perspective.

The Analytics component systematically correlates collected data, to detect anomalies and non-compliance events. Those findings will help to highlight the relevant facts and improve the post-mortem identification of security events to be extracted as evidence. The adoption of a continuous auditing approach, against a set of predefined policies, helps detect and mitigate possible threats.

The FCA works in a hybrid fashion: cyber–physical system anomalies are handled by the corresponding domain-specific IDS, which by their turn feed the FCA with the analysis outcomes. For Operational Technology (OT) systems above IEC62443 [62] layer 2 (comprising Area Supervisory Control components such as SCADA servers or historian databases) and IT systems, the FCA is able to receive direct event feeds (from probes and log sources), later to be normalized and preprocessed. Anomaly detection in the scope of the FCA framework takes advantage of both the post-processed CPS IDS data as well as from the OT and IT feeds.

Next, we discuss each individual module of the FCA framework.

3.2. Data acquisition

The Data Acquisition Module (DAM) is able to ingest a large amount of data, for instance from application logs, AAA services, ICT security tools (e.g., anti-virus), internal personnel activities, physical access control logs (door switches and surveillance cameras), maintenance activities (physical and logical systems), interactions with third-parties (e.g., general documents, emails) and incident logs (e.g. ICT trouble tickets). This module can eventually be decoupled across separate gateway/edge instances to optimize load distribution.

The integration of sources from third parties is implemented by customizing a data adapter provided by an interface. This way, the DAM manages the incoming data from heterogeneous sources $S_j^{(\Gamma)} \in S$, with their own schemas Γ. This results in discrete events $E^{(S_j)}$, collected from probes P_{E_i} assigned to sources S_i. Those events contain a set of independent attributes or features, where i is the ith event and k denotes the total number of features in a discrete event $E_i^{(S_j)}$.

The incoming flow of raw data goes through a pipeline producing events described relying in a common schema — and later persisted into a Data Lake (DL) for further analysis. The set of tuples $(a, v)_k \in \{1..n\}$ are composing parts of events E_i, as denoted by:

$$E_i = \bigcup_{i=1}^{n} (a, v)_k \tag{1}$$

A tuple includes n attributes a and their corresponding values v, and is addressed by an index or key k, according to:

$$\{(a, v)\}_{k \in \{1..n\}} \in E_i^{(S_j)}, a \in A, v \in T \tag{2}$$

A value v of any type $t \in T$, scalar or not, is assigned to an attribute $a \in A$, where A represents the set of all existing attributes. Common attributes include, for example, the provenance and time when the events were created.

This module also provides an interface to add new data sources S_{i+1}. A parsing function $P()$ converts the incoming data events E_i into a discrete event N_i, according to:

$$P : E_i \rightarrow N_i \tag{3}$$

The adoption of a common schema for normalized events N_i facilitates the integration processes between the internal components and external third party applications. This schema can be either followed or extended by third parties.

The DAM takes the role on managing the data ingestion $I()$, processing the received events E_i from the different sources S_j to be parsed $P()$, validated $V()$ and normalized $N()$, according to:

$$I = [N \circ V \circ P]() \tag{4}$$

This ingestion function $I()$ of a given event E_i results in a normalized event $N_i^{(1)}$, such as:

$$N_i^{(1)} = I(E_i) \tag{5}$$

The parsing function $P()$ extracts the features as attributes and values in a set of tuples $(a, v)_k$. These attributes and values come from a raw event $E_i^{(S_j)}$, received from system S_j, and are transformed into an event $E_i^{(\Gamma_{S_i})}$, following the native schema of the corresponding data source Γ_{S_i}:

$$P : E_i^{(S_j)} \rightarrow E_i^{(\Gamma_{S_i})} \tag{6}$$

The validation function $V()$ checks whether an event $E_i^{(\Gamma_{S_i})}$ follows the schema Γ_{S_i} associated with its source system S_i.

$$v_i = V(E_i^{(\Gamma_{S_i})}, L), v_i \in \{0, 1\} \tag{7}$$

In case this check fails, that event is discarded and recorded as a validation failure V, for further analysis, as denoted by:

$$V = \bigcup_{i=1}^{N} v_i \tag{8}$$

The normalization function $N()$ maps an event $E_i^{(\Gamma_{S_i})}$ in schema Γ_{S_i} to an event $N_i^{(\Gamma_{S_i})}$ following the general schema Γ of the framework, according to:

$$N : E_i^{(\Gamma_{S_i})} \rightarrow N_i^{(\Gamma)} \tag{9}$$

Raw events E_i are subject to a set of functions, including parsing $R()$, validation $V()$ and normalization $N()$ into a common format $N_i^{(1)}$. Finally, the events collected at the first stage $N_i^{(1)}$ are collected as part of the set of normalized events N_{DA} in DAM, according to:

$$N_{DA} = \bigcup_{i=1}^{N} N_i^{(1)} \tag{10}$$

To keep overall performance, probes receiving data are decoupled from the rest of the components in the framework. Decoupling components from data sources makes it easier to support the assigned workloads, even if the data sources are delivered at a high pace (e.g. Packet Capture Process (PCAP) or data logs from a large number of endpoints). To that aim, the incoming events are stacked in a queue Q, waiting for the availability of component C_i.

The performance and availability of the framework is impacted by overloaded components. Decoupling the intertwined asynchronous communication between the core functions can be achieved by queuing the underlying events $E_i^{(\Gamma)}$. This offers the opportunity to shift in time the future workloads. This queue function $Q()$ interfaces the coupled components according to:

$$Q_{C_{i-1},C_i} = Q(C_{i-1}, C_i), E_i^{(\Gamma)} \in Q_{C_{i-1},C_i} \tag{11}$$

Since different components along the ingestion stream along may have different ingestion rates, queuing becomes necessary.

The rate of received data Δ results from counting events ξ for a given period of time Δt, according to:

$$\xi_{\Delta t} = \frac{(\xi_{T_1} - \xi_{T_0})}{\Delta t} \tag{12}$$

The output rate $\theta_{\Delta t}$ represents the difference between ingested (ξ) and consumed rates (θ) in a certain period of time Δt, yielding:

$$\theta_{\Delta t} = \frac{(\theta_{T_1} - \theta_{T_0})}{\Delta t} \tag{13}$$

3.3. Domain processor

The Domain Processor Module (DPM) transforms and loads ingested events. This module gathers the functions for filtering $F()$, indexing $I()$, enriching $E()$, and aggregating $A()$. At this stage, all the events' metadata can be extracted into a keyword index for later support of queries. Therefore, the second stage of the normalization process over events $N_i^{(1)}$ to $N_i^{(2)}$ takes place in this module. The domain processor module may be decoupled across distributed gateway/edge instances to optimize load distribution, either as a separate entity, or consolidated together with Data Acquisition module instances.

As previously stated, this module receives a flow of normalized events $N_i^{(1)}$, resulted from the pipeline (as a sequence of stages) provided by the DAM. Thus, the processing function $R()$ applies the sequence of the above functions $\{A, E, I, F\}$, as denoted by:

$$R = [A \circ E \circ I \circ F]() \tag{14}$$

The processing function $R()$ receives as input events of the first stage $N_i^{(1)}$ to output the second stage events $N_i^{(2)}$, according to:

$$R : N_i^{(1)} \rightarrow N_i^{(2)} \tag{15}$$

For the sake of simplicity, from now on N_i will denote the result of the second normalization process $N_i^{(2)}$.

Next, we discuss each of the four functions that compose the DPM.

Filtering provides filtering capabilities $F()$, supported by rules R_i, to check if events $E_i^{(j)}$ in domain $\{0, 1\}$ should be accepted to be forwarded to the next component, as denoted by:

$$F : E_i^{(j)} \rightarrow \{0, 1\} \tag{16}$$

A filter F denotes a logical disjunction of n Boolean rules R_i, according to:

$$F = R_1 \vee R_2 \vee \cdots \vee R_n = \bigvee_{i=1}^{n} R_i \tag{17}$$

A rule R_i may either denote a conjunction of positive or negative disjunctions of specific attribute levels, as defined by:

$$R_i = \bigwedge_{a_k \in A_{R_i}} S_k \tag{18}$$

$A_{R_i} \subset A$, denotes a subset of attributes in rule R_i. $\bigvee$ and $\bigwedge$ represent the Boolean algebra operators OR and AND. The S_k refers to a logical statement about the kth attribute $a_k \in A_{R_i}$. Thus, S_k is composed by two distinct parts, according to:

$$S_k = n_k \bigvee_j l_k^j \tag{19}$$

The first part is the disjunction of level values with l_k^j the jth level of the attribute a_k. The second part is the parameter $n_k \in [1, \neg]$, which allows negating (logical operator NOT) the disjunction when set to $\neg$. The user enters specific rules specifying the levels l_k^j and the parameters n_k.

$$R_i = \bigwedge_{a_k \in A_{R_i}} (n_k \bigvee_j l_k^j) \tag{20}$$

The filtering function $F()$ can target the values of those attributes through the use of rules. Thus, a set of disjunction rules allows to discard some of those events. In case either attributes or values do not match the logical definition of the filter, F is TRUE. In that case the event is blocked.

Indexing supports the analysis of data in real-time (e.g., by means of visualization tools, along with associated analysis capabilities). Function $I()$ indexes the events being ingested to improve the performance of answering to queries $q_i \in Q$. It takes as input a set of attributes $\{a\} \subset N_i$ and data in $N \subseteq DL$ as A stored in DL to return a subset of k tuples $\{(a, v)_i\}$, as denoted by:

$$I = I(A, N), (a, v)_i \in I \tag{21}$$

Enrichment relies in the $E()$ function to add contextual features (attribute and value $(a, v)_i$) to the events being ingested (e.g. associate geolocation data to an IP-address), to increase their usefulness. This way, the correlation of data outside of the framework will be avoided, enhancing the usage of the collected data to a larger number of use cases. It takes as input a set of n existing attributes $\{a_i\}_{i=1}^{n}$ as A to produce m new features $\{(a, v)_i\}_{i=1}^{m}$ as C^m by extracting the values v contained in external enrichment sources S_E, through the use of function $M_e()$, according to:

$$C^m = \{(a, v)_i\}_{i=1}^{m} \tag{22}$$

and

$$C^m = M_e(A, S_E), (a, v)_i \in A \tag{23}$$

The enrichment function $E()$ adds new m attributes to the set of existing attributes of the normalized event N_i, as denoted by:

$$E : N_i^{(n)} \rightarrow N_i^{(n+m)} \tag{24}$$

The resulting event is denoted by $N_i^{(n+m)}$, gathering $C^{(m)}$ features from an enrichment source S_E according to:

$$N_i^{(n+m)} = E(N_i^n, C^m, S_E) \tag{25}$$

An example of an enrichment function is adding the geographical location to the event, based on the already existing IP address attribute. Other examples are the inference of User and Asset elements or even the related DNS information (e.g. retrieved from a "Whois" command).

Finally, *aggregation* corresponds to the $G()$ function, that summarizes the data from events N_i, according to a given set of grouping attributes $H \subset A$ applied to a set of operations functions $k_i() \in K$ (e.g. sums, minimum, maximum or average). The resulting events G are persisted into the DL for further analysis $G \subset DL$. The aggregation function $G()$ is denoted by:

$$G = G(N, H, K), G_i \in N \tag{26}$$

with $G_i \in G$ and $\{(a_a, v)_i\}_{i=1..n}$ as the set of summarized attributes and $\{(a_s, v)_j\}_{j=1..m}$ as the set of summarizing functions as attributes and their resulted values from functions $k_i() \in K$, according to:

$$G_i = \{(a, v)_i\}_{i=1..n} \cup \{(a, v)_j\}_{j=1..m} \tag{27}$$

This allows, for instance, to create indicators such the number of events per second (grouped per category).

3.4. Cyber–physical IDS

The Cyber–physical IDS is an external module that identifies threats at CI level, by means of fast path processing. This would typically be the IDS systems already deployed at the Critical Infrastructure that feed the FCA framework.

Similarly to the other data sources producing events $N_i \in N$, this module contributes with discrete events $N_i \in N_{IDS}$ as result of the application of the classification function $S()$, to be persisted into the central data lake, according to:

$$N_{IDS} = S(N), N_{IDS} \subset DL \tag{28}$$

Where $N_{IDS} \subset DL$ will be used for further analysis.

3.5. Data Lake (DL)

The DL is the central storage to the framework. It relies on the $\psi()$ function for gathering and persisting the normalized events $N_{DP} \in DL$, as denoted by:

$$N_{DP} = \psi(N), N_{DP} \subset DL \tag{29}$$

Where events N result both from normalized events resulting from the processing in the DPM and from events received directly from the IDS:

$$N = \{N_{IDS}, N_{DP}\} \tag{30}$$

Other sources that feed the Data lake include the intermediary functions, such as generated monitoring alerts Y, trust and reputation data T, and aggregation data G.

Thus, the DL maintains the collected data from all existing different sources producing events, including the enriched data, according to the following data sources:

$$DL = \{N, Y, T, G\} \tag{31}$$

3.6. CI business rules

The Critical Infrastructure's Business Rules $R_i \in R$ can be used to describe the policies $R_i \subseteq P$ supposedly adopted by the target organization — so that the FCA can assess the effective compliance with those rules, by scoring incoming events $N_i \in N_{DP}$ against rules $_i \in R$. As a result of this evaluation $Ev(R_i, N) \in \{0, 1\}$, alerts A_{R_i} can be generated according to adjustable thresholds.

For example, an organization can specify a set of policies $\{p\}_i \in P$ to enforce the access control to resources from users within their own organizational unit, and report any login attempt violating that policy. Another example is the physical access control to facilities. In that regard alerts can be triggered in case doors of a given department are opened out of the authorized time range.

3.7. Analytics

The Analytics Module provides the capabilities to classify events N_{DL} and detected anomalies from events in DL, adding a new attribute for this purpose (a_{n+1}, k). Event-contained features provide the data inputs to classify $K()$ anomalies K or get the knowledge that can help future investigations.

The function $K()$ is used to classify all the events in the data lake, flagging them as normal or anomalous, based on the insights resulting from the correlation of the N_{DL} normalized events in the DL.

For example, by applying a classification learning model $ML^{(k)}$ to a normalized event N_i results in a binary classification $k \in \{0, 1\}$ a new feature (a_{n+1}, k) to be included in a event $k_i \in K$. Function $K()$ provides the intelligence to classify event threats, accordingly to:

$$k_i = K(N_i, ML^k), k_i \in K, \forall N_i \in N_{DL} \tag{32}$$

In this classification example, $k_i \in \{0, 1\}$, $\{0\}$ corresponds to an event marked as non-abnormal, while $\{1\}$ corresponds to an anomaly. Similarly to the enrichment process, a new feature (a_{n+1}, k) in an event N_i containing the result from such classification k is introduced. To this effect, the features $\{a_i\}_{i=1}^n$ of second stage events $N_i^{(2)}$ are updated to include an additional attribute $n \leftarrow n+1$, resulting in a third stage event $N_i^{(3)}$.

The events classified as anomalies in $N_b \in B$ can help to understand the chain of events (e.g. since the moment an attack has started until it has finished). These events are chronologically ordered according to their creation timestamp a_t. The reconstruction of the path of events t_b results from the set of sequenced events classified as anomalies following a numerical order $N_1, N_2, N_3 \in N_{DL}$ in the sequence $N_1 \rightarrow N_2 \rightarrow N_3$. That implies that the absolute times when they are produced are consistently ordered so that $t_1 < t_2 < t_3$.

This component relies on models enabled by Machine Learning (ML) algorithms to automate the classification of anomalous events. These classification algorithms help to identify and mitigate the impact of potential threats to the integrity, confidentiality or availability of the CI. Each classification function $K()$ applies the set of learning models $ML_j \in ML$, where j denotes the jth learning model. These models distil and extract the insights from the ingested normalized events. Selected features contained in the normalized event N_i will be used as inputs to the function $K()$, for classification purposes.

In this paper we describe these ML algorithms from a generic and abstract point a view, since the framework is agnostic regarding which specific algorithms are used. Nevertheless, in previous works we explore specific ML approaches in the scope of FCA, including for instance the combination of K-means with XGBoost [63] and the usage of decision-trees for policy and audit compliance purposes [64].

The framework includes two different sets of learning models, one for training purposes $ML^{(t)}$ and another for classification $ML^{(k)}$, as denoted by:

$$ML = \{ML^{(t)}, ML^{(k)}\} \tag{33}$$

Models are gradually trained according to more recent data to be used for classification purpose. To that aim, a deployment function D is used to transfer the learning model ML_i from the training set $ML_i^{(t)}$ to the classification set $ML_i^{(k)}$, accordingly to:

$$D : ML_i^{(t)} \rightarrow ML_i^{(k)} \tag{34}$$

3.7.1. Forensic analytics

The Forensics Analytics component includes the functions for supporting the investigation process of examiners on extracting evidence for helping them to reconstruct the path and timeline of events to identify and bookmark anomalies and non-conforming situations previously classified by the Analytics Module (AM).

Moreover, this component will provide the control mechanisms over the information being exchanged, read, and processed by different entities along the investigation chain. Thus, in that regard, it will be important to record examiner activities on handling digital forensic evidence along the investigation chain. The adoption of a standardized approach on sharing evidence will help investigators to collaborate in the identification of the root cause of events (e.g. criminals for crimes committed in different jurisdictions). Evidence structure follows a common Forensic Schema Γ_A along the investigation chain. Proof of evidence provenance is achieved by including a specific attribute in events $a_p \in N_i$.

This component provides support to query data from $q_i \in Q$ gathering a set of n rules:

$$q_i = \{R_j\}_{i=j}^{n} \tag{35}$$

The forensic analysis function $Z()$ finds out the events $Z \subset N_{DL}$ in DL to be extracted and digitally signed with a hash H. This cryptographic hashing H will help to trace the events to their sources and check their authenticity and integrity, accordingly to:

$$Z = Z(N_{DL}, Q, H), Z_i \in Z, Z_i \in N_{DL} \tag{36}$$

3.7.2. Audit compliance

The Audit Compliance component is in charge of checking security, company policies or adherence to standard practices as a way to improve overall protection.

Such an assessment can result from laws emanated by external regulations or standards the CI operator has chosen or needs to follow. The CIs can also dictate their internal regulatory rules and decides when specific business policies (R_c), comprising corporate laws, policies, plans, and procedures, should be followed.

This component offers continuous assessment and compliance check capabilities. It ensures due diligence, certification, and stakeholder security by checking whether the CI meets the regulatory requirements and follows the industry guidance. In that regard, the audit compliance function $W()$ checks whether compliance rules R_c are being violated or not against the DL events N_{DL}. Such an assessment will help to identify the non-compliance rules $W \subset R_c$, as denoted by:

$$W = W(N_{DL}, R_c), r_i \in W \tag{37}$$

This component provides the means, for instance, for computing the scores quantifying the risk level. Therefore, by identifying the non-conformance rules will be possible to score the security risk level function $R_c()$ from rules R_c:

$$R_l : W \rightarrow r_l, r_l \in \mathbb{N} \tag{38}$$

The auditor can trigger auditing tasks upon request or enable and schedule them to be executed at regular intervals. This component will automatically notify the auditors Ac with the auditing results, according with the specific roles Rl assigned in the platform.

3.7.3. Data visualization

In the final stage of forensics and compliance auditing activities the Data Visualization component provides visualization capabilities $V()$, reporting, summarizing and displaying the collected data in a suitable, transparent and simple to use way. The function $V()$ fits the visualization data $v_i \in V$ from events N_{DL} in the DL through the use of a set of queries $q_i \in Q$ and defining the summarizing attributes A_s and functions $f_i() \in F$, according to:

$$V = V(N_{DL}, Q, A_s, F), V \subset N_{DL} \tag{39}$$

Dashboard panels are the visualization artifacts highlighting the information in V. The panels V present the information in different ways (e.g. histogram with events being received according to their type of requests, logical representation of the infrastructure). That information is ready to be exported and support further analysis by third-party tools.

3.7.4. Real time search

The Real Time Search Component relies in function $J()$ for querying data from the DL with low latency to achieve near real-time results. The function $J()$ runs in parallel with other functions along the ingesting pipeline. It returns the set of events $J \subset DL$ for a given period Δt, according to a set of queries $q_i \in Q$ and the indexing data I, as denoted by:

$$J = J(I, Q, \Delta t), J \subset DL \tag{40}$$

The indexing function $I()$ offers the capabilities to function $J()$ to achieve results in near real-time. The collected information is also made available to actors through user interfaces and third-parties, by means of integration components (e.g. to detect intruders before they have access to resources).

3.8. Trust and reputation indicators

This component integrates the work of Caldeira et al. [65] for computing sets of trust and reputation indicators $K_{\Delta t}$ for a period of time Δt. The function $K_{\Delta t}$ relies in the trust function $T()$ to compute those indicators in a given period from events N_{DL} in DL, as defined by:

$$K_{\Delta t} = T(N_{\Delta t}) \tag{41}$$

The trust function $T()$ includes the reference levels of risk (Rl_i) and the current measured risk levels (Ml_i), in order to detect deviations and highlight threats.

Therefore, $K_{\Delta t}$ takes function $T()$ for averaging the measured risk level Ml_i and the received risk alert level Rl_i, computed for n events $N_{\Delta t}$, according to:

$$T(N_{\Delta t}) = 1 - \frac{\sum_{i \in N_{\Delta t}} \Phi(Ml_i, Rl_i)}{n} \tag{42}$$

Finally, the function $R_t()$ computes the risk level between CIs for each event N_i. This function considers current events (N_i) and past events stored in the data lake (N_{DL}), with $T_{N_i}^{(i,j,s)}$ denoting the cascade risk level between critical infrastructures i and j for service s and for the ith event N_i, as defined by:

$$T_{N_i}^{(i,j,s)} = R_t(N_i, N_{DL}, T_{N_i}^{(i,j,s)}) \tag{43}$$

and

$$R_t(N_i, N, T_{N_i}^{(i,j,s)}) = \frac{(N-1)\phi T_{N_i}^{(i,j,s)} + K_{N_i}}{(N-1)\phi + 1} \tag{44}$$

The function Φ is a discrete function and value k applies penalties to higher differences between levels of risk (Rl_i) and the current measured risk levels (Ml_i), as denoted by:

$$\Phi(Ml_i, Rl_i) = \left|\frac{(Ml_i - Rl_i)}{4}\right|^{k}, k \in R^{+} \tag{45}$$

The concept of aging is implemented by applying more weight ϕ to recent events. Trust and reputation can be regarded as a set of continuous indicators to be persistent $t_i \in T$, according to:

$$T = Tr(N_{\Delta t}, T^{(i,j,s)}), t_i \in T \quad (46)$$

3.9. Business rule compliance monitoring

The Monitoring Module continuously monitors data to assess the compliance of observed events with predefined rulesets, to assure the CI remains compliant with adopted user and business rules. It also allows operators to identify non-compliant events $N_i \in M$, by checking the rules $r_{mi} \in R_m$. The monitoring function $M()$ checks whether user and business rules R_m are being met by evaluating them against the events in DL N_{DL} for a given period of time Δt. The function $M()$ assesses the CI business rules $r_i \in R$ supporting auditing compliance activities from events in Data Lake $N_i \in DL$, and trust and reputation risk alert levels Rl_t to classify eventual threats and triggering alerts to the operators, according to:

$$M = M(N_{DL}, \Delta t, R_m), N_i \in M \quad (47)$$

The set M gathers the events $N_i \in M$ deemed as non-compliant with rules R_m, which trigger non-compliance event alarms.

3.10. Orchestration

The Orchestration Module provides the means for coordinating and automating the management of the various framework modules.

3.11. Actors and roles

Departing from a conventional IDS role, the FCA constitutes a system that persists relevant data, providing the means for forensics experts to search and identify relevant events, also enabling continuous auditing of organizational processes and procedures — for these purposes, knowledge extraction mechanisms allow for the definition of rules that can be used for both (semi-)automated forensics analysis and for auditing compliance purposes. Thus, it can be considered that operators and security analysts are the main actors of the framework.

Forensic and auditing experts are security analysts searching for facts to be extracted as evidence. They investigate the trail of events to identify the root cause of relevant events and involved systems and criminals. Along their investigation, they run *ad hoc* queries in the available data in the data lake. At the end, collected evidence are reported to stakeholders to help them to organize and take responsive actions.

Operators can define the level of criticality associated with non-compliance events for each rule. Those criticality levels are evaluated either from events being monitored or from triggered alerts. CI operators rely on the Data Visualization Component capabilities and make use of user interfaces depicting summarized data (e.g. graphically displayed in dashboards).

Permission levels granted to each actor are role-based. To increase granularity, each actor is associated with a set of role(s) for each platform module, and each module role is associated with a set of (allowed) functions.

The security analyst perceives the level of threat from the incoming events highlighted by the Monitoring Module. These actors take their investigations activities by introducing *ad hoc* queries to understand the chain of events.

The actions and decisions taken by these actors are logged as self-accountable mechanisms of the framework. Therefore, all these actions are potential inputs to resolve future conflicts such as data privacy violation allegations and studying examiner investigative styles for learning and training purposes.

4. Proof-of-concept implementation

A Proof-of-Concept (PoC) of the proposed FCA framework has been implemented and deployed as part of the Intrusion and Anomaly Detection System (IADS) of the ATENA H2020 project [66].

The ATENA project combines new anomaly detection algorithms and risk assessment methodologies within a distributed environment, to provide a suite of integrated market-ready ICT networked components and advanced tools embedding innovative algorithms [67]. It provided the reference scenarios, use cases and testbeds needed to validate our research, allowing to collect valuable data from realistic testbeds in order to evaluate the prototype frameworks throughout experimental use cases scenarios.

Within the IADS, the FCA platform provides the mechanisms to persist a broad spectrum of digital pieces of evidence obtained from multiple data sources, for forensics analysis and policy conformity checks. The implementation of the FCA was supported by an Elasticsearch stack (Elasticsearch, Logstash, and Kibana) as an efficient manual and semi-automatic searching tool to cope with the complexity and massive amount of available data.

Within ATENA, the IADS module is responsible for monitoring the underlying CI environment using distributed probes. Moreover, the distributed probes will generate events for suspicious activity, which will be processed before being sent to the upper layers of the ATENA architecture to be classified [68].

Fig. 2 illustrates the ATENA cyber-security architecture with their modules. Beyond the FCA, it includes: different types of probes, from conventional network and host probes to IACS field-specific probes; a Domain Processor per scope, backed by a Message Queuing system; a distributed SIEM, for support of streaming and batch processing; and a Data Lake, where all the data is stored [68].

5. Evaluation and discussion

In this section we evaluate and discuss the FCA framework. We start this evaluation with a general description of the experimental setup (Section 5.1), followed by the analysis of the results obtained for data ingestion (Section 5.2) and query handling performance evaluation (Section 5.3). Next, we present two Use Cases that highlight the potential of the proposed framework for detecting cyberattacks: the Password Cracking Use Case (Section 5.4) and the User Account Control Bypass Use Case (Section 5.5).

It should be noted that the proposed FCA framework is intended to be agnostic regarding which specific analytics algorithms are used. In fact, each specific deployment scenario will probably call for different algorithms. For this reason, this evaluation focuses more on performance than accuracy. Nonetheless, in previous works we have already accessed the accuracy of specific ML algorithms in the scope of FCA, such as Decision Trees [64] and K-Means with XGBoost [63].

5.1. Experimental setup

As already mentioned, the implemented PoC uses the Elasticsearch open-source full-text search engine technology for storage and query capabilities. Additionally, the FCA platform forms a cluster composed of a set of nodes deployed as Docker containers.

The infrastructure for the experiments used a VMware Hypervisor to host a single Virtual Machine (VM) with 8 vCPUs (Intel Xeon Gold 5120 CPU @ 2.20 GHz), 64 GB of RAM and 48G of SSD storage. This VM supported the Elasticsearch nodes, deployed as Docker containers from Elasticsearch 6.2.4 images and the clients requesting ingestion and computation capabilities from these nodes.

The Elastic Search nodes were constrained in terms of CPU and memory (through the use of mem_limit: 2G, memswap_limit: 2G cpus: 0.5). The ingestion producers generate events to be ingested into a given node with a bulk operation. They were deployed as containers

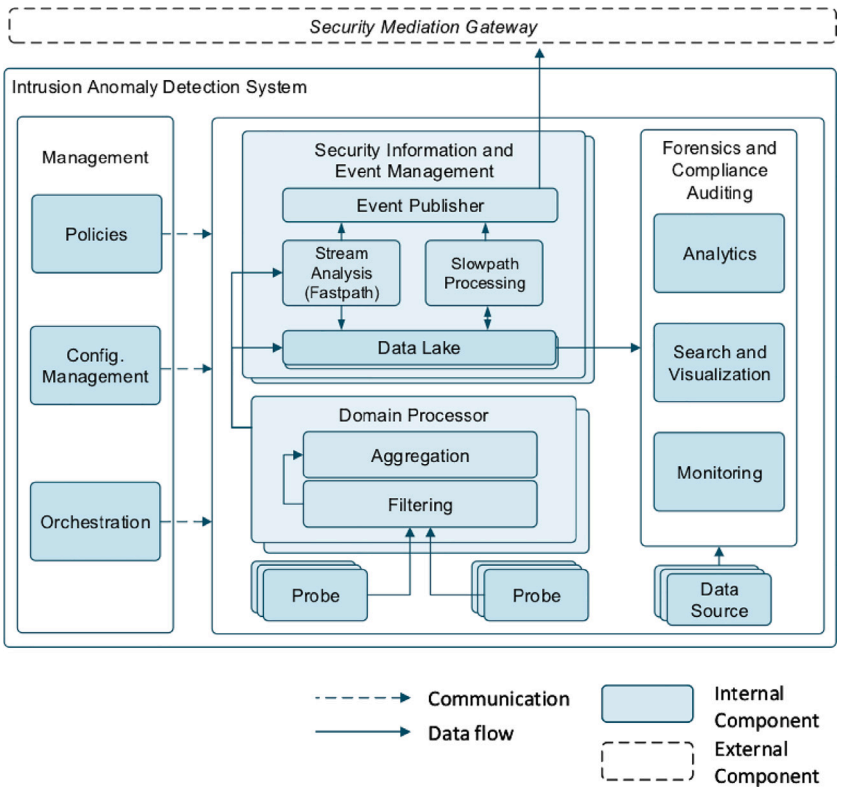

Fig. 2. ATENA cyber-security architecture [68].

from custom Docker images (python:3.8-slim-buster). The containers were limited to 50 percent in terms of CPU and 2 GB of RAM (Docker settings: –memory 2G, –memory-swap 2G and –cpus 0.5). Similarly, also the computation clients were deployed as containers from custom Docker Python images taking the responsibility on querying a given node, in this case constrained with 25% percent in terms of CPU and 2 GB of RAM.

The experiments use synthetic events, generated by the clients, with a payload of 160 bytes, formed by two fields: “any” and “timestamp”, the latter corresponding to the moment the event was created. Events are ingested by the FCA platform through a unique bulk request from the client. Listing 1 depicts the client python source code feeding the ingestion of events into the “river” index and “tweet” type of the Elasticsearch cluster. It receives the identification of the block for events and the number of events to be generated in a loop and stored in an array. Finally, they are pushed into a single endpoint of the cluster, in a single bulk operation, and the time to ingest them is recorded.

Listing 1: Client Source Code

```
from datetime import datetime
from elasticsearch import Elasticsearch
from elasticsearch import helpers
import time
import sys

start_time = time.time()
block=int(sys.argv[1])
records=int(sys.argv[2])
es = Elasticsearch("http://172.27.221.159:9204/")
actions = [
  {
    "_index": "river",
    "_type": "tweet",
    "_id": block+j,
    "_source": {
        "any":"data" + str(j),
        "timestamp": datetime.now()}
  }
  for j in range(0, records)
]
helpers.bulk(es, actions)
elapsed_time = time.time() - start_time
```

The purpose of this setup was to assess the performance of the framework, by measuring its latency on ingestion and computing operations. To this purpose, the FCA framework was evaluated with different combinations of Elasticsearch nodes (3, 5 and 20 nodes) and shards (1 and 5 primary shards, and 1 to 5 replica shards). Different combinations of feeding clients (also designated as producers) were also used. Within the Elasticsearch terminology, a shard is a logical block that stores data and indexes. Shards can be replicated by different nodes along the cluster, with each one being classified as a primary or replica. A primary shard is the block assigned with the role of the primary storage for data, while replicas maintain a copy. Replicas can be added or removed anytime, to scale out/in computing queries (by default, Elasticsearch would create one primary shard and one replica for every index).

To assess the FCA ingestion performance, 10 producers generate 10 Million events to be ingested evenly, distributed by all cluster nodes. Regarding FCA computation performance, a client evenly queries the cluster nodes with a simple operation (counting the 10 Million ingested events), using the DSL Query language (based on JSON). Our analysis used mostly the maximum time it took to complete the operations, even though minimum and average time are also reported.

Table 1
Time to ingest 10 million events (ms).

N	PS	R	Minimum	Maximum	Average
3	1	1	207 384	217 280	212 591
3	3	1	177 157	180 524	179 492
3	5	1	183 895	188 718	186 804
3	5	2	176 575	181 951	180 244
3	5	3	183 568	186 511	185 543
3	10	1	252 972	255 232	254 256
5	1	1	2 636 090	2 792 717	2 751 700
5	3	1	1 319 386	1 383 388	1 350 092
5	5	1	1 334 385	1 409 409	1 369 947
5	5	2	1 283 605	1 362 303	1 329 752
5	5	3	1 359 796	1 419 684	1 385 052
5	10	1	1 208 413	1 225 788	1 219 440
10	1	1	2 590 074	2 706 697	2 680 579
10	3	1	863 380	891 132	875 575
10	5	1	871 322	901 407	886 429
10	5	2	863 380	881 599	873 379
10	5	3	864 986	910 689	891 984
10	10	1	856 180	893 589	874 900

Table 2
Time to compute 10 million events (ms).

N	PS	R	Minimum	Maximum	Average
3	1	1	6	18	11
3	3	1	8	108	43
3	5	1	9	182	40
3	5	2	6	60	15
3	5	3	9	27	15
3	10	1	16	119	41
5	1	1	6	19	11
5	3	1	8	62	19
5	5	1	8	31	16
5	5	2	7	63	17
5	5	3	9	35	17
5	10	1	86	493	274
10	1	1	14	212	122
10	3	1	9	14	12
10	5	1	6	25	15
10	5	2	10	28	16
10	5	3	9	14	11
10	10	1	116	681	366

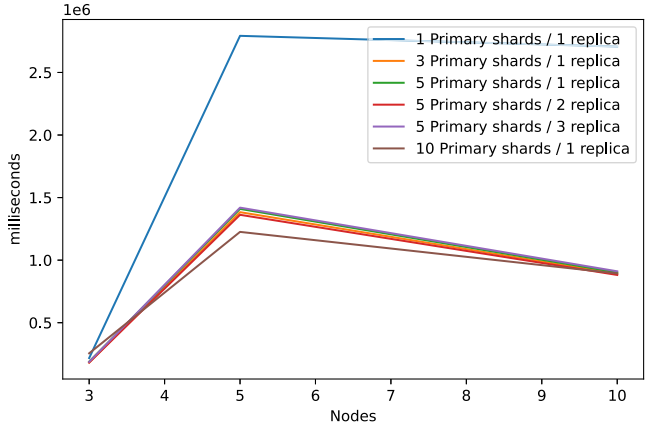

Fig. 3. Max ingestion time (ms).

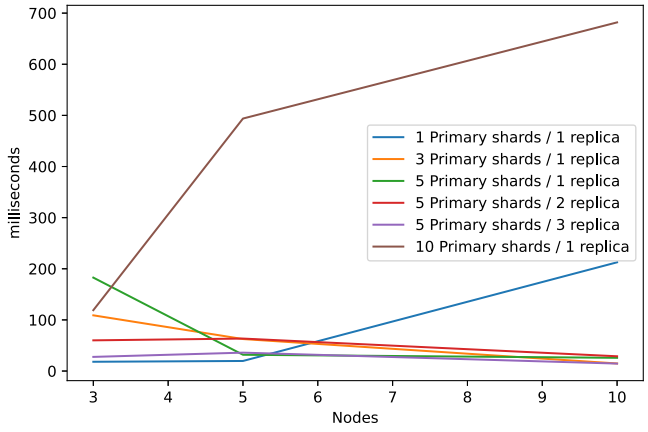

Fig. 4. Maximum computation time (ms).

5.2. Ingestion performance

A first set of experiments measured the time it takes to ingest 10 Million events when using eighteen distinct setups. First, we adjusted the number of nodes (N) to 3, 5 or 10. Moreover, we also adjusted the number of primary shards (PS) (1, 3, 5 and 10) and replicas (R) (1, 3 and 5).

Table 1 summarizes the obtained results, for each setup, when ingesting 10 Million events. Fig. 3 depicts the maximum ingestion time (in milliseconds) when running the 10 clients for different cluster sizes and under different settings of primary shards and replicas. It should be noted that, for this specific experiment, there were no significant differences observed between maximum, average and minimum times.

The achieved results highlight the scalability of the cluster. Most notably, it can be observed that the injection latency decreases as the number of nodes increases from 5 to 10. For all different settings, including the primary shards (3, 5, and 10) and replicas (1, 2, and 3), results denote an improvement in the ingestion performance when the cluster size increases from 5 to 10 nodes. On the other hand, the cluster reveals a poor performance scenario when ingesting before the 5 node threshold is reached. Maybe it can be explained with the trade-off between the size and overhead caused by management activities of the cluster.

5.3. Computational Performance

A second set of experiments measured the framework performance when computing the 10 Million events previously ingested along the first experiment. For this purpose, 10 different clients are running the same query simultaneously, distributed evenly across the available nodes. The query requires checking the different ID of all events (cf. Listing 2).

Listing 2: Query Counting Records

```
{"size":0,
    "aggs":{
        "DistinctWords":{
            "cardinality":{
                "field":"id"
    }   }   }
}
```

The eighteen aforementioned setups were also used for this experiment, therefore considering different cluster sizes, different numbers of primary shards and different numbers of replicas. Table 2 summarizes the observed results.

Figs. 4, 5, 6 depict the maximum, minimum and average computation time for each setup. Unlike the previous experiment, larger variations between maximum, minimum and average times were observed in a few situations, namely when using one primary shard and one replica that keeps a low minimum computation time for different cluster sizes.

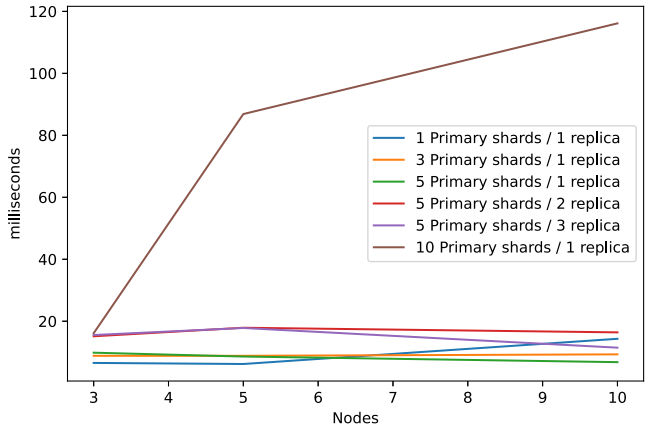

Fig. 5. Minimum computation time (ms).

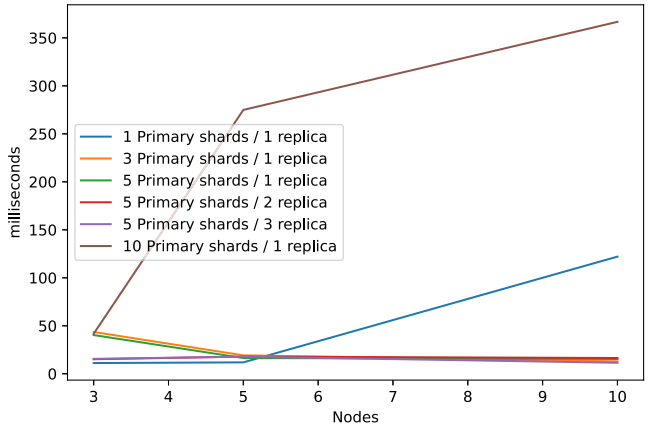

Fig. 6. Average computation time (ms).

These results highlight the scalability of the cluster, except in the case of 1 and 10 primary shards. All other settings (3 and 5 primary shards and 1, 2, and 3 replicas) denote an effective improvement in ingestion performance when increasing the size of the cluster: from 3 to 5 nodes, and from 5 to 10 nodes. This also holds when different settings are considered for primary shards (3 and 5) and replicas (1, 2, and 3).

The results also denote that defining a cluster with one primary shard is a bad option when the objective is to scale it in a distributed environment. Another observation is that the computation performance improves when the number of shards (primary and replica) is higher than the number of nodes.

5.4. The password cracking use case

Many attacks start with an unauthorized entry into the network, and then evolve to scouting and exploitation of weaknesses in other nodes of the network, using horizontal movement [69]. Quite often, such attacks involve a progression chain, with multiple stages and different sources. Independent ML models might recognize such attacks in different sources, albeit in a stateless way, lacking the overall perspective about the attack progression and the relationship between multiple stages.

In this section we show how the FCA framework can be used to improve the overall detection process, by recognizing the different stages of those attacks through the analysis of different sources. More specifically, this is done by combining the classification of heterogeneous data from different sources to recognize multistage attacks and the relationship between them.

A rule R_i recognizing an attack at stage i can be applied along n stages, with $i \in \{t_0, t_1, \ldots, t_n\}$. Optionally, an additional rule $R_{(i-1,i)}$ may be used to check if there is a connection between stages $i-1$ and i. This way, it is possible for instance to flag a "password crack" attack at stage S_1, preceded by a "recognition" attack at stage S_0 within a predefined time window (for instance 7 days: $t_{(0,1)} = (t_1 - t_0) <= 7$).

According to the FCA framework, rules can target the classified events $E_i \in E$ in the DL. Moreover, each ML model $ML_i^{(S_j)} \in ML$ can be attached to a distinct source $j \in \{S_0, S_1, \ldots, S_n\}$. Each of those rules R_j can be defined as a tuple encompassing three sub-rules $(R_{i-1}, R_i, R_{(i-1,i)})$. The first sub-rule R_{i-1} evaluates an event attribute $a_j \in E_i$ in a first source S_0, with name "attack" and value "true". The second sub-rule R_i also checks a previous classified attack in a source S_0. Finally, a third sub-rule $R_{(i-1,i)}$ evaluates the precedence between those two events, for instance checking if they (E_{i-1} and E_i) differ less than 7 days.

To demonstrate this Use Case, we used the TON_IoT datasets [70], which comprise heterogeneous data sources collected from telemetry datasets of IoT services, Windows and Linux-based datasets, as well as datasets of network traffic. Events are labeled as normal or as generated under the different kind of attacks, including: Scanning, Password cracking, Denial of Service (DoS), Distributed DoS, Ransomware, Backdoor, Injection, Cross-site Scripting (XSS), Password cracking and Man-In-The-Middle (MITM).

The Scanning Attack (and also Reconnaissance and Probing Attacks) corresponds to the first stage of the cyber kill chain model. This kind of attacks usually tries to discover active IP addresses and open ports in the targeted network, in order to prepare the following stages of attack.

In our first application scenario, the capabilities of the FCA Framework are leveraged by using Elasticsearch technology. First to ingesting the different TON_IoT datasets to different indexes, and later to apply the rules aiming to recognize the various stages of attacks supported by their classified events. In this scenario, it is possible for instance to check if a password cracking attack occurred (within the "Modbus" TON_IoT dataset), and if it was preceded by a scanning attack (in the "Windows 7" TON_IoT dataset). According to data contained in those datasets, a password cracking attack occurred on April 27, and it was possible to realize it was preceded by a scanning attack that occurred two days before.

A first rule R_i checks if events are classified as "password" attacks. For that purpose, an Elasticsearch query Q_0 (type:"password" and date ≥ "27-Apr-19") was defined, targeting the "modbus" dataset, stored in a specific "modbus" Elasticsearch index, as defined in Listing 3 query, which returned 4665 results.

Listing 3: Password Cracking Elasticsearch Query to Modbus index R_i

```
GET /modbus/_search
{
  "query":{
    "bool":{
      "must":[
        {"match":{"type":"password"}}
      ],
      "filter": [
        {"range":{"date":{"gte":"27-Apr-19"}}}
      ]
    }
  }
}
```

A second Elasticsearch query Q_1 (type :"scanning" and @timestamp ≥ "2019-04-25") materializes the rule $R_{(i-1)}$, which checks if current the "Password Cracking" was preceded by a "Scanning attack". This query checks another source ("windows 7" dataset collected into a specific Elasticsearch index "windows7"), as defined in Listing 4, which returned 71 results.

Table 3
Sequence of stage queries to recognize a cracking password attack scenario.

Stage	(1) Type	(2) @timestamp	Results
1	Password	27–Apr–19	4665
2	Scanning	2019–04–25	71

Listing 4: Scanning Phase Elasticsearch Query to Windows 7 index $R_{(i-1)}$

```
GET /windows7/_search
{
  "query":{
    "bool":{
      "must":[
        {"match": { "type":"scanning"}}
      ],
      "filter":[
        {"range":{"@timestamp":
            {"gte":"2019-04-25"}}}
      ]
    }
  }
}
```

Finally, the rule $R_{(i-1,i)}$ has been implicitly materialized in the previous Elasticsearch queries, namely as part of the condition @timestamp ≤ "2019-04-25" in $R_{(i-1)}$ and date ≥ "27-Apr-19" in R_i.

All the Elasticsearch queries filtering the type and date of the attack, as well as the number of results along the two stages, are summarized in Table 3.

5.5. The user account control bypass attack use case

Currently, forensic activities focus mostly on servers and network security. Usual approaches include IDS, firewalls and proxies, while the regulatory compliance focuses on analyzing logs from assets. On the other hand, attacks on desktop endpoints are usually underestimated. Nevertheless, it is important to realize that a significant number of recent exploits attacks start at desktop endpoints.

Actions taken by attackers in the Windows endpoints Operating Systems (OS) leave back a set of logs, which raise the chances of detection. In this Section, we describe how the proposed framework can be used to detect such attacks by recognizing the stages of an User Account Control (UAC) Bypass attack by means of analyzing logged actions at the endpoints.

The UAC is a Windows OS capability that allows programs to run at administrator level. When the programs introduce changes, the user is informed and prompted for confirmation to elevate their privileges. Despite this, in some circumstances, certain Windows programs are allowed to elevate privileges without authorization by the UAC.

In this experimental work, we started by mimicking the usual steps of this kind of attack by injecting the usually generated logs into the endpoint OS, in order to later determine if it is possible to detect such attacks. According to MITRE ATT&CK [71], a UAC Bypass attack comprises the following steps: USB compromise, persistence, theft of credentials, and their reuse by installing a backdoor in a local account. Finally, the attacker may want clear the logs.

In a first stage S_0, the attacker tries to compromise an endpoint OS by exploiting its weaknesses — for instance through the use of virtual keyboard USB devices or Powershell scripts. As a result, the attacker may compromise the endpoint system by installing a Command and Control (C2). To that aim, a PowerShell C2 can be launched from a virtual keyboard USB device, a corresponding DriverFrameworkd-UserMode (2001) event log being recorded. Other event logs can also be recorded at the system level, in the case a Powershell script is executed (4100) or a new service (System 7045) is installed.

Table 4
Sequence of stage queries to recognize a user account control bypass attack.

Stage	(1) Event_Id	(2) Timestamp	Results
1	2001	2019–04–24T01:42:58	1
2	4698	Stage 1 Timestamp	1
3	4624	Stage 2 Timestamp	1
4	4720	Stage 3 Timestamp	1
5	7045	Stage 4 Timestamp	1

In the second stage S_1, the backdoor is installed, which may trigger flagging events at the Windows endpoint OS, such as new services (System 7045), scheduling tasks (System 4698), or registry modification (System 4657).

In the third stage S_2, the attacker steals the credentials to later reuse them. This can trigger system windows event logs such as successful login (System 4624) or failed login (Security 4625).

In the fourth stage S_3, the backdoor is installed through the use of a local account. This is achieved by including a new local user into the Administrators group in a critical system. As a consequence, the following Windows security events may be triggered: adding new users (Security 4720) or adding the user to the local group (Security 4732).

Finally, in the last stage S_4, the attacker may try to clear their logs from endpoints and critical systems. Those actions can be logged as a new service (System 7045), scheduling a new task (System 4698), or even trying to modify the registry (System 4657).

For each one of the previous actions, a log event is recorded into the Windows endpoint OS, by the use of a Powershell script. Next, the WindowsLogBeat agent installed in the host pushes those logs into the Elasticsearch. There, those logs are parsed by a grok regular expression in the Logstash component (DAM), processed along the pipeline by the DPM and, finally, stored into Elasticsearch (DL).

The operator can define a rule R_j as a tuple of stage rules $(R_i, R_{i-1}, R_{(i,i-1)})$, supported by Elastic queries. These rules try to recognize the typical sequence of stages R_j of an UAC Bypass attack, with $j \in \{0, 1, 2, 3, 4\}$. Stage queries aim to recognize a specific stage of the attack chain and the (possible) connection to the previous stage actions.

An example of this query, filtering the event_id (<EVENT_ID>) and @timestamp (<TIMESTAMP>) fields, is provided in Listing 5. Table 4 summarizes obtained results.

Listing 5: Query Windows Logs Events

```
GET /windowslogs/_search
{
  "query":{
    "bool":{
      "must":[
        {"match": { "type":"windowseventlog"}}
      ],
      "filter":[
        {event_id:<EVENT_ID>},
         {"range":{"@timestamp":
            {"gte":"<TIMESTAMP>"}}}
      ]
    }
  }
}
```

Table 5
Summary of data and function symbols.

Module	Name	Type	Description
Data acquisition	E_i	Data	Event
	a_i	Data	Event attribute
	$\xi_{\Delta t}$	Data	Counting events
	S_i	Data	Data source
	$\theta_{\Delta t}$	Data	Output rate
	I()	Function	Data ingestion
	N()	Function	Normalization
	Q()	Function	Queue
	P()	Function	Parsing
	V()	Function	Validation
Domain Processor	C^m	Data	Enriched event
	S_E	Data	Enrichment source
	H	Data	Grouping attributes
	K	Data	Operations
	Q_i	Data	Queries
	R_i	Data	Rule
	S_k	Data	Rule statement
	G	Data	Summarizing data
	E()	Function	Enrichment
	M()	Function	Enriching from sources
	F()	Function	Filtering
	I()	Function	Indexing
	k_i()	Function	Operation
	G()	Function	Summarizing
CI Business Rules	P	Data	Policies
	A_{R_i}	Function	Alert from a rule
	Ev()	Function	Auditing evaluation
Data Lake	DL	Data	Data Lake
	N	Data	Normalized events
	ψ()	Function	Persist
Analytics	K	Data	Anomalies
	ML_i	Data	ML model
	K()	Function	Classification anomalies
	D()	Function	Deployment
Forensics Analytics	q_i	Data	Query
	Γ_A	Data	Forensic schema
	Z	Data	Forensic analyzed event
	Z()	Function	Forensic analysis
	H	Function	Hash
Audit Compliance	W	Data	Non-compliance rules
	W()	Function	Audit compliance
Data Visualization	V()	Function	Visualization
Real time Search	Δt	Data	Period of time
	J()	Function	Real Time search
Trust & Reputation	K	Data	indicators
	Ml_i	Function	Measured risk level
	Rl_i	Data	Risk level alert
	Rt()	Function	Risk level
	Φ()	Function	Risk level penalty
	T()	Function	Trust
Business Rule	R_c	Data	Business policies
	M	Data	Non compliant rules
	M()	Function	Monitoring

6. Conclusions and future work

This work intends to provide a foundation to build up improved solutions addressing the current and future requirements in the field of FCA. It formally presents a framework unifying the FCA capabilities to protect IACS, describing their function blocks and functions, as well as their internal, input, and output communications.

This framework is able to manage a significant amount of heterogeneous data moving under intense flows, from the moment it is ingested until insights are extracted, by relying on distributed computing resources to achieve a high-performing system providing results near real-time. The key components of the framework were explored in terms of their ability to scale to specifically cope with the volume and speed at which data is produced.

The experimental evaluation, which encompassed several experiments to evaluate the performance of a cluster under different settings for ingestion and computing workloads, demonstrated the suitability of the proposed framework to cope with FCA requirements at scale. Moreover, this experimental evaluation also showed how the proposed framework can be useful to detect typical chains of attacks.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was partially funded by the FCT–Foundation for Science and Technology, I.P./MCTES through National Funds (PIDDAC), within the scope of Centre for Informatics and Systems of the University of Coimbra (CISUC) Research and Development Unit, under Grant UIDB/00326/2020 and Project UIDP/00326/2020. It was also co-funded by FEDER, via the Competitiveness and Internationalization Operational Program (COMPETE 2020) of the Portugal 2020 framework, in the scope of Project Smart5Grid (POCI-01-0247-FEDER-047226) and by Project “Agenda Mobilizadora Sines Nexus” (ref. No. 7113), supported by the Recovery and Resilience Plan (PRR) and by the European Funds Next Generation EU, following Notice No. 02/C05-i01/2022, Component 5 - Capitalization and Business Innovation - Mobilizing Agendas for Business Innovation. Furthermore, would also like to thank the Research Center in Digital Services (CISeD) and the Polytechnic of Viseu for their kind support.

References

[1] L. Martin, 5Th annual edition of cyber defense magazine - 2017 predictions, 2017, https://www.tradepub.com/free-offer/cyber-defense-magazine--2017-predictions, visited on 2017-04-01.
[2] ATENA, D2.1 state of art, 2016, https://www.atena-h2020.eu, visited on 2016-12-01.
[3] E. Morioka, M. Sharbaf, Cloud computing: Digital forensic solutions, in: International Conference on Information Technology-New Generations, in: 12, Las Vegas, 2015, pp. 589–594.
[4] D.R. Rani, G. Geethakumari, An efficient approach to forensic investigation in cloud using VM snapshots, in: 2015 International Conference on Pervasive Computing (ICPC), 2015, pp. 1–5, http://dx.doi.org/10.1109/PERVASIVE.2015.7087206.
[5] NIST, Guide to integrating forensics techniques into incident response, 2017, http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-86.pdf, visited on 2017-06-01.
[6] K. Sindhu, B. Meshram, Digital forensic investigation tools and procedures, in: International Journal of Computer Network and Information Security, in: IJCNIS, 2012, http://dx.doi.org/10.5815/ijcnis.2012.04.05.
[7] R. Hunt, J. Slay, Achieving critical infrastructure protection through the interaction of computer security and network forensics, in: 2010 Eighth International Conference on Privacy, Security and Trust, 2010, pp. 23–30, http://dx.doi.org/10.1109/PST.2010.5593243.
[8] G.M. Mohay, Computer and Intrusion Forensics, Artech House, 2003.
[9] A. Pauna, K. Moulinos, M. Lakka, J. May, T. Tryfonas, Can We Learn from SCADA Security Incidents, White Paper, European Union Agency for Network and Information Security, Heraklion, Crete, Greece, 2013.
[10] D. Kushner, The real story of stuxnet, Ieee Spectr. 50 (3) (2013) 48–53.
[11] J.T. Langill, Defending against the dragonfly cyber security attacks, Retrieved 11 (2014) 2015.
[12] M. Fillinger, M. Stevens, Reverse-engineering of the cryptanalytic attack used in the flame super-malware, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2015, pp. 586–611.
[13] ICS-CERT, Cyber-attack against ukrainian critical infrastructure, 2016, https://ics-cert.us-cert.gov/alerts/IR-ALERT-H-16-056-01, visited on 2016-12-01.

[14] R. Khan, P. Maynard, K. McLaughlin, D. Laverty, S. Sezer, Threat analysis of BlackEnergy malware for synchrophasor based real-time control and monitoring in smart grid, in: 4th Int'L Symposium ICS & SCADA Cyber Security Research. BCS, 2016, pp. 53–63.

[15] D. Quick, K.-K.R. Choo, Impacts of increasing volume of digital forensic data: A survey and future research challenges, Digit. Investig. 11 (4) (2014) 273–294.

[16] C.F. Tassone, B. Martini, K.-K.R. Choo, Visualizing digital forensic datasets: A proof of concept, J. Forensic Sci. (2017).

[17] J. Koven, E. Bertini, L. Dubois, N. Memon, InVEST: Intelligent visual email search and triage, Digit. Investig. 18 (2016) S138–S148.

[18] A.R. Javed, W. Ahmed, M. Alazab, Z. Jalil, K. Kifayat, T.R. Gadekallu, A comprehensive survey on computer forensics: State-of-the-art, tools, techniques, challenges, and future directions, IEEE Access 10 (2022) 11065–11089.

[19] F. Casino, T.K. Dasaklis, G. Spathoulas, M. Anagnostopoulos, A. Ghosal, I. Borocz, A. Solanas, M. Conti, C. Patsakis, Research trends, challenges, and emerging topics in digital forensics: A review of reviews, IEEE Access (2022).

[20] S. Rizvi, M. Scanlon, J. Mcgibney, J. Sheppard, Application of artificial intelligence to network forensics: Survey, challenges and future directions, IEEE Access 10 (2022) 110362–110384.

[21] N.G. Ganesh, N.M. Venkatesh, D.V.V. Prasad, A systematic literature review on forensics in cloud, IoT, AI & blockchain, Illum. Artif. Intell. Cybersecur. Forensics (2022) 197–229.

[22] V. Roussev, G. Richard, Breaking the performance wall: The case for distributed digital forensics, in: Proceedings of the 2004 Digital Forensics Research Workshop, Vol. 94, 2004.

[23] Y. Xie, D. Feng, Z. Tan, J. Zhou, Unifying intrusion detection and forensic analysis via provenance awareness, 61, Future Gener. Comput. Syst. (2016) 26–36.

[24] C. Valli, SCADA Forensics with Snort IDS, CSREA Press, 2009.

[25] P. Turner, Unification of digital evidence from disparate sources (Digital Evidence Bags), Digit. Investig. (ISSN: 1742-2876) 2 (3) (2005) 223–228, http://dx.doi.org/10.1016/j.diin.2005.07.001, [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287605000575.

[26] P. Turner, Selective and intelligent imaging using digital evidence bags, Digit. Investig. (ISSN: 1742-2876) 3 (2006) 59–64, http://dx.doi.org/10.1016/j.diin.2006.06.003, [Online]. Available: http://www.sciencedirect.com/science/article/pii/S174228760600065X, The Proceedings of the 6th Annual Digital Forensic Research Workshop (DFRWS '06).

[27] R. Eaglin, J. Craiger, Data sharing and the digital evidence markup language, in: 1st Annual GJXDM Users Conference, Atlanta, GA.(Not Peer Reviewed), 2005.

[28] S.S. Lee, T.-S. Park, S.-U. Shin, S.-K. Un, D.-W. Hong, A new forensic image format for high capacity disk storage, in: Information Security and Assurance, 2008. ISA 2008. International Conference on, IEEE, 2008, pp. 399–402.

[29] B.N. Levine, M. Liberatore, DEX: Digital evidence provenance supporting reproducibility and comparison, Digit. Investig. (ISSN: 1742-2876) 6 (2009) S48 – S56, http://dx.doi.org/10.1016/j.diin.2009.06.011, [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287609000395, The Proceedings of the Ninth Annual DFRWS Conference.

[30] E. Casey, G. Back, S. Barnum, Leveraging cybox™ to standardize representation and exchange of digital forensic information, Digit. Investig. 12 (2015) S102–S110.

[31] A. Aminnezhad, A. Dehghantanha, M.T. Abdullah, A survey on privacy issues in digital forensics, Int. J. Cyber-Secur. Digit. Forensics 1 (4) (2012) 311–324.

[32] R. Verma, J. Govindaraj, G. Gupta, Data privacy perceptions about digital forensic investigations in india, in: Ifip International Conference on Digital Forensics, Springer, 2016, pp. 25–45.

[33] P.R. Grammatikis, P. Sarigiannidis, E. Iturbe, E. Rios, A. Sarigiannidis, O. Nikolis, D. Ioannidis, V. Machamint, M. Tzifas, A. Giannakoulias, et al., Secure and private smart grid: The spear architecture, in: 2020 6th IEEE Conference on Network Softwarization (NetSoft), IEEE, 2020, pp. 450–456.

[34] P.R. Grammatikis, P. Sarigiannidis, A. Sarigiannidis, D. Margounakis, A. Tsiakalos, G. Efstathopoulos, An anomaly detection mechanism for IEC 60870-5-104, in: 2020 9th International Conference on Modern Circuits and Systems Technologies (MOCAST), IEEE, 2020, pp. 1–4.

[35] ISO 27001 Security, ISO27k toolkit, ISMS auditing guideline, version 2, 2017, 2017, http://www.iso27001security.com/html/27007.html, visited on 2017-06-01.

[36] ISO/IEC, ISO/IEC 27002:2022 - information security, cybersecurity and privacy protection — Information security controls, 2022, https://www.iso.org/standard/75652.html, visited on 2023-02-17.

[37] P. Mell, T. Grance, The NIST definition of cloud computing (NIST SP 800-145), 2011, http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, visited on 2016-12-01.

[38] ISO/IEC, ISO/IEC 27019:2017. Information technology—Security techniques—Information security controls for the energy utility industry, 2017, https://www.iso.org/standard/68091.html, visited on 2023-02-17.

[39] ISA SECURE, Establishment of isasecure Japanese scheme and publication of isasecure embedded device security assurance certification program specifications in Japan, 2013, http://www.isasecure.org/en-US/News-Events/Establishment-of-ISASecure-Japanese-Scheme-and-Pub.

[40] E.A. Morse, V. Raval, PCI dss: Payment card industry data security standards in context, Comput. Law Secur. Rev. 24 (6) (2008) 540–554.

[41] IEC, IEC 62443 - IEC security for industrial automation and control systems - Part 4-2: Technical security requirements for IACS components, 2019, https://webstore.iec.ch/publication/34421, visited on 2023-02-13.

[42] K. Fisler, S. Krishnamurthi, L.A. Meyerovich, M.C. Tschantz, Verification and change-impact analysis of access-control policies, in: Proceedings of the 27th International Conference on Software Engineering, 2005, pp. 196–205.

[43] G.-J. Ahn, H. Hu, J. Lee, Y. Meng, Representing and reasoning about web access control policies, in: 2010 IEEE 34th Annual Computer Software and Applications Conference, IEEE, 2010, pp. 137–146.

[44] K. Arkoudas, R. Chadha, J. Chiang, Sophisticated access control via SMT and logical frameworks, ACM Trans. Inf. Syst. Secur. 16 (4) (2014) 1–31.

[45] K.W. Ullah, A.S. Ahmed, J. Ylitalo, Towards building an automated security compliance tool for the cloud, in: 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, 2013, pp. 1587–1593.

[46] F. Doelitzscher, Security Audit Compliance for Cloud Computing (Ph.D. thesis), Plymouth University, 2014.

[47] N. Bjørner, K. Jayaraman, Checking cloud contracts in microsoft azure, in: International Conference on Distributed Computing and Internet Technology, Springer, 2015, pp. 21–32.

[48] IBM, IBM Security QRadar SIEM, 2022, https://www.ibm.com/qradar/security-qradar-sieml, visited on 2022-01-17.

[49] AWS, Security at scale: Logging in AWS, 2022, https://d1.awsstatic.com/whitepapers/compliance/AWS_Security_at_Scale_Logging_in_AWS_Whitepaper.pdf?did=wp_card&trk=wp_card, visited on 2022-01-17.

[50] S. Majumdar, T. Madi, Y. Wang, Y. Jarraya, M. Pourzandi, L. Wang, M. Debbabi, Security compliance auditing of identity and access management in the cloud: Application to OpenStack, in: 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, 2015, pp. 58–65.

[51] K. Kent, S. Chevalier, T. Grance, H. Dang, Guide to integrating forensic techniques into incident response, NIST Spec. Publ. 10 (2006) 800–886, visited on 2017-06-01.

[52] Gartner, Magic quadrant for security information and event management, 2016, https://www.gartner.com/doc/3406817/magic-quadrant-security-information-event, visited on 2017-01-01.

[53] G. González-Granadillo, S. González-Zarzosa, R. Diaz, Security information and event management (siem): Analysis, trends, and usage in critical infrastructures, Sensors 21 (14) (2021) 4759.

[54] Securonix, Securonix security analytics platform, 2016, http://www.securonix.com/security-intelligence, visited on 2016-12-01.

[55] IBM, SIBM security intelligence with big data, 2016, http://www-03.ibm.com/security/solution/intelligence-big-data, visited on 2016-12-01.

[56] RSA, RSA Security Analytics, 2016, http://www.emc.com/collateral/data-sheet/security-analytics-overview-ds.pdf, visited on 2016-12-01.

[57] LogRhythm, LogRhythm security analytics, 2016, https://logrhythm.com/products/security-analytics, visited on 2016-12-01.

[58] Pravail, Pravail security analytics, 2016, https://www.pravail.com, visited on 2016-12-01.

[59] Alienvault, Alienvault: A integrated solution with real-time threat intelligence, 2016, http://www.alienvault.com, visited on 2016-12-01.

[60] Cisco, OpenSOC: Big data security analytics framework, 2016, http://opensoc.github.io, visited on 2016-12-01.

[61] Apache Metron, Apache metron: Real-time big data security, 2016, http://metron.incubator.apache.org, visited on 2016-12-01.

[62] IEC, IEC 62443 - IEC technical specification - industrial communication networks - network and system security - Part 1-1: Terminology, concepts and models, 2017, https://webstore.iec.ch/preview/info_iec62443-1-17Bed1.07Den.pdf, visited on 2017-03-01.

[63] J. Henriques, F. Caldeira, T. Cruz, P. Simes, Combining K-means and xgboost models for anomaly detection using log datasets, Electronics (ISSN: 2079-9292) 9 (7) (2020) http://dx.doi.org/10.3390/electronics9071164, [Online]. Available: https://www.mdpi.com/2079-9292/9/7/1164.

[64] J. Henriques, F. Caldeira, T. Cruz, P. Simes, An automated closed-loop framework to enforce security policies from anomaly detection, Comput. Secur. (ISSN: 0167-4048) 123 (2022) 102949, http://dx.doi.org/10.1016/j.cose.2022.102949, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404822003418.

[65] F. Caldeira, S. T., M.E. Varrette S., S. P., B. P., K. D., Trust based interdependency weighting for on-line risk monitoring in interdependent critical infrastructures, in: International Journal of Secure Software Engineering, in: 4, IGI Global, 2013.

[66] L. Rosa, T. Cruz, M.B. de Freitas, P. Quitério, J. Henriques, F. Caldeira, E. Monteiro, P. Simões, Intrusion and anomaly detection for the next-generation of industrial automation and control systems, Future Gener. Comput. Syst. 119 (2021) 50–67.

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

[68] ATENA, D4.1 requirements and reference architecture for the cyber-physical IDS, 2017, https://www.atena-h2020.eu/project-documentation/, visited on 2017-07-01.

[69] N. Jaswal, Hands-on Network Forensics: Investigate Network Attacks and Find Evidence using Common Network Forensic Tools, Packt Publishing Ltd, 2019.

[70] N. Moustafa, A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets, Sustainable Cities Soc. 72 (2021) 102994.

[71] MITRE, Bypass user account control, 2017, [Online]. Available: https://attack.mitre.org/techniques/T1548/002/, visited on 2017-08-01.