frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 05 January 2023
DOI 10.3389/fcomp.2022.1020946

OPEN ACCESS

EDITED BY
Haibin Zhu,
Nipissing University, Canada

REVIEWED BY
Mohammad Jabed Morshed Chowdhury,
La Trobe University, Australia
Damian Tamburri,
Eindhoven University of Technology, Netherlands

*CORRESPONDENCE
Remo Pareschi
✉ remo.pareschi@unimol.it

SPECIALTY SECTION
This article was submitted to Software, a section of the journal Frontiers in Computer Science

RECEIVED 16 August 2022
ACCEPTED 05 December 2022
PUBLISHED 05 January 2023

CITATION
Di Pilla P, Pareschi R, Salzano F and Zappone F (2023) Listening to what the system tells us: Innovative auditing for distributed systems.
Front. Comput. Sci. 4:1020946.
doi: 10.3389/fcomp.2022.1020946

Listening to what the system tells us: Innovative auditing for distributed systems

Piergiuseppe Di Pilla[1], Remo Pareschi[1,2]*, Francesco Salzano[1] and Federico Zappone[1,2]

[1]Stake Lab, University of Molise, Campobasso, Italy, [2]BB-Smile Srl, Rome, Italy

Introduction: In recent years, software ecosystems have become more complex with the proliferation of distributed systems such as blockchains and distributed ledgers. Effective management of these systems requires constant monitoring to identify any potential malfunctions, anomalies, vulnerabilities, or attacks. Traditional log auditing methods can effectively monitor the health of conventional systems. Yet, they run short of handling the higher levels of complexity of distributed systems. This study aims to propose an innovative architecture for system auditing that can effectively manage the complexity of distributed systems using advanced data analytics, natural language processing, and artificial intelligence.

Methods: To develop this architecture, we considered the unique characteristics of distributed systems and the various signals that may arise within them. We also felt the need for flexibility to capture these signals effectively. The resulting architecture utilizes advanced data analytics, natural language processing, and artificial intelligence to analyze and interpret the various signals emitted by the system.

Results: We have implemented this architecture in the DELTA (Distributed Elastic Log Text Analyzer) auditing tool and applied it to the Hyperledger Fabric platform, a widely used implementation of private blockchains.

Discussion: The proposed architecture for system auditing can effectively handle the complexity of distributed systems, and the DELTA tool provides a practical implementation of this approach. Further research could explore this approach's potential applications and effectiveness in other distributed systems.

KEYWORDS

distributes systems, log auditing, log analysis, NLP, blockchain, Hyperledger Fabric

1. Introduction

Distributed systems have been spreading rapidly in recent years (Zheng et al., 2017), and the emergence of Distributed Ledger Technologies (DLTs) such as blockchains have strongly contributed to this trend. These technologies find a wide range of possible applications in areas such as the Internet of Things (IoT), healthcare, supply chain management, energy, genomics, fintech, insurance, automotive, etc.

(Dorri et al., 2017; Bottoni et al., 2020; Carlini et al., 2020; Fosso Wamba et al., 2020; Motta et al., 2020; Wang and Su, 2020). As a consequence, there is an ongoing strengthening of development frameworks such as *Ethereum*, *Hyperledger*, *EOSIO*, *Corda*, *Waves*, *Quorum* etc. which are constantly adding new features.

The trend is explained by the ability of DLT to provide a high degree of security, compared to classical systems, by encrypting and decentralizing data, aspects that are both paramount for the development of decentralized applications (*dApps*). Data security is a crucial issue for information systems in general. Over time, numerous tools have been developed for data protection and monitoring system access, including auditing, which provides a wealth of security-related information. In particular, log auditing extracts information about the operations carried out and the conditions in which they took place and keeps track of their timelines. Logs are, therefore, essential for analyzing system behavior under normal and abnormal conditions. Indeed, while the normal case provides a history of the operations carried out, the anomalous one helps identify system errors and detect vulnerabilities, thus preventing cyberattacks.

However, compared to development frameworks, distributed systems auditing tools lag, especially when it comes to blockchains. The more established ecosystems, such as Bitcoin and Ethereum, have their log analysis tools (Crystal-team, 2019; Knownsec-404, 2019). Yet, there is a lack of a standardized tool that can be integrated with most frameworks and is endowed with real-time monitoring capabilities. Existing tools designed for cloud and decentralized systems (Kufel, 2016) are not easy to integrate with development frameworks for blockchain applications and are not up to the challenges regarding complete log auditing of blockchain systems (Cangemi and Brennan, 2019). We started from these premises to define a general methodology and architecture for constructing a log auditing system suitable for blockchain and distributed ledger requirements. The architecture is defined in such a way that it can be applied to all systems with distribution characteristics, i.e., characterized by a multiplicity of nodes on which independent agents operate, the only prerequisite being that there is the possibility of tracing actions by writing logs to files. We then instantiated this architecture by using it for the design and development of a universal log analysis tool, which takes the name of *DELTA* for Distributed Elastic Log Text Analyzer, aimed at analyzing logs of activities on most of the existing development frameworks for distributed and non-distributed systems—a versatility which is made possible thanks to the use of the *Docker Engine* (Reis et al., 2022) and the *stack ELK* (*Elasticsearch, Logstash, Kibana*) integrated *via Filebeat*, which have a proven history of applications in log collection (Alekseev et al., 2018; Bavaskar et al., 2020).

For this purpose, we use the *Docker Engine* as a bridge for collecting logs between the analyzer and distributed systems. Thanks to *Docker*, it is, in fact, possible not only to integrate completely different systems but also to analyze the logs produced through the *ELK* stack. This stack makes it possible to efficiently control log collection methods by accessing *Docker containers*. Furthermore, *Filebeat* takes care of managing log collection methods in real-time, and *Logstash* enables automatic log insertion into the *Elasticsearch* database, which in turn supplies data to *Kibana* for immediate viewing through a customizable graphical interface.

The developed tool couples traceability with textual analysis of the traced logs through Natural Language Processing (NLP). NLP is used to perform, upon the text within logs produced by the *Docker containers*, three types of analysis: keyword extraction, classification, and sentiment analysis. Keywords are extracted through two different models: the more precise *KeyBERT*, based on *BERT* (Bidirectional Encoder Representations from Transformers), and the more versatile *YAKE!* (Campos et al., 2020). As regards log classification, the choice fell on the *Zero-Shot facebook/bart-large-mnli* developed by Meta (formerly Facebook), which works without requiring data outside the text. The idea of *Zero-Shot* models is to analyze and classify data that are also completely different from those with which the training was carried out using the methodology for statistical inference described in Lewis et al. (2019). Finally, sentiment analysis is performed through *VADER* (Valence Aware Dictionary and sEntiment Reasoner), an open-source analysis tool based on rules for extracting sentiment using dictionaries. All log collection, analysis, and classification processes occur in real-time, enabling interaction with resource monitoring processes. This way, it is possible to focus log analysis on the security problems and undertake mitigating actions as needed.

Structure of the paper. The remaining part of the article is organized as follows: Section 2 describes the proposed generic architecture for log auditing of distributed systems. Section 3 describes *DELTA* and the components used for its implementation. Section 4 describes the application of DELTA to Hyperledger Fabric, the most popular platform for private blockchains. Section 5 describes related work. Section 6 concludes the paper and gives indications on future work.

2. Methodology

Our systems design choices are rooted in Grounded Theory (GT) (Glaser and Strauss, 2017), which has a long-standing reputation in software engineering (Stol et al., 2016). In this approach, a series of systematic steps ensure that the theory emerges from the data (hence, grounded). In our case, this

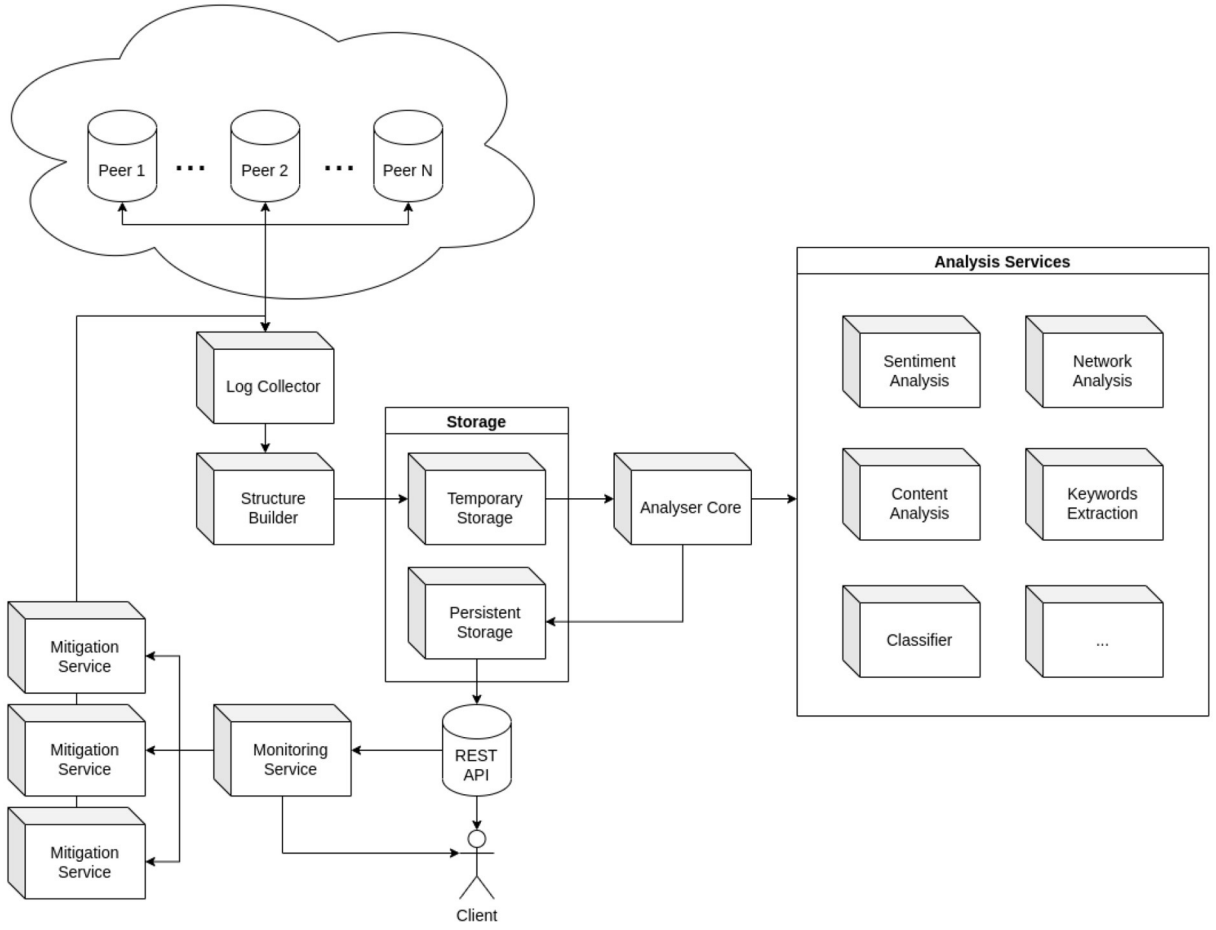

FIGURE 1
Generic log auditing architecture for distributed systems.

implied the iterative verification of the logs characteristics and the applicable analysis methods. We then verified the possibility of automating the log collection and analysis workflow. This led to the construction of the *DELTA* tool described in the next section. Finally, we subjected *DELTA* to a process of abstraction to obtain the architecture presented in this section, which is compatible with various implementation choices, of which *DELTA* is but one. From the point of view of the narrative, we offer these results in reverse compared to how they were achieved, thus going from the general architecture towards one of its instances. This presentation structure is consistent with the overall philosophy underlying GT: on the one hand, the theory (that is, in our case, the system architecture) emerges bottom-up from the collection, observation, and experimentation with data; on the other, once stabilized, the theory is usable top-down to derive as many of its instances as wanted.

Distributed systems are typically characterized by having an arbitrary number of nodes performing all network operations. Often the nodes do not share the same resources; on the contrary, they have entirely separate environments to guarantee their independence. As a result, when in operation, they generate logs independently. Collecting and analyzing the logs from all records is a cumbersome operation if approached with traditional methods. Indeed, to perform a log analysis capable of bringing valid results in good time, it will be difficult or even inapplicable to analyze each of the logs separately. The alternative methodology we provide relies primarily on the fact that, before being analyzed, log data are centralized by using the elastic stack, as we will illustrate in the following sections. To implement it adequately, we devised an architecture aimed at the system's distributed nature. A representation of its main components and their role within the system is given in Figure 1. It is worth noting that this strategy can be re-applied to any distributed system that writes logs into streams. The proposed architecture functions with docker by default. Still, as we will discuss in Section 3.1, it can work with any available sources.

2.1. Log collector

The first step is to collect the logs, thus dealing with the problems arising from the independence of the nodes so that logs of different nodes do not necessarily coincide. It would, therefore, not be realistic to collect just the logs produced by a single node or a subset of nodes as there is no guarantee that they reflect the general opinion of the network: for example, in a system based on consensus mechanisms such as a blockchain, one or more nodes can express views on a transaction that differ from those of the other nodes, hence letting some anomalies go unnoticed. The Log Collector is the component of the architecture that handles this crucial step and is also the only one that simultaneously interacts with the set of nodes in its entirety to collect all their logs and other relevant data, such as the specific metadata of the structure of the distributed system, which can range from simple peer identifiers to a copy or verification of the security certificates used to communicate with nodes. Once data collection is complete, the Log Collector forwards everything to the Structure Builder.

2.2. Structure builder

The second component of the system is aimed at creating a structure for effectively managing all collected data to provide efficient data exchange between the components while avoiding duplications and redundancies. The main aspects relevant for selecting the structure used for storage and exchange are the quality and consistency of collected data and the technologies available to build up the system (DBMS, applied analysis techniques, etc.). Once the initial modeling is done, the data goes into Storage.

2.3. Storage

The Storage mechanism is divided into two sub-components: Temporary and Persistent Storage. Temporary Storage is a queue-and-cache system into which the Structure Builder inserts data collected and structured but not yet analyzed. The data held temporarily are then pipelined into Persistent Storage, which also receives the results of their analyses carried out by the Analyzer Core while they were still in Temporary Storage. Finally, data can move externally from Persistent Storage through integration *via* REST APIs. Note that the implementation of the two Storage components can build upon a single DBMS (Elasticsearch, MongoDB, etc.) or two or more, giving preference, for example, to systems designed for caching for Temporary Storage (e.g., key-value databases) and more structured systems if needed for Persistent Storage (e.g., relational databases).

2.4. Analyser core and analysis services

The Analyzer Core is the central component of the architecture, with the task of analyzing the data in Temporary Storage according to whatever techniques are deemed appropriate and can therefore be designed and implemented with complete flexibility regarding which analyses to perform. NLP techniques are natural candidates since the logs mainly consist of messages whose text was initially provided at the development time of the executed code. The main objective of the analysis services is to extract implicit information from the collected data, thus providing further data by which it is possible to query the logs. For example, keyword extraction identifies the most significant words from the log text to use them for keyword-based searches on Persistent Storage. Generally speaking, the Analyzer Core aims to obtain data for effective problem solving. In this regard, in addition to extracting keywords, Sentiment Analysis, Content Analysis, and Classification techniques can also be integrated to obtain useful filtering information, and Network Analysis can be used to identify significant relationships between nodes.

2.5. REST API extensibility

Data collection and analysis are vital aspects of an auditing system. Still, they would turn out null and void in the absence of suitably engineered information access, as provided by the REST API module that enables accessing the Persistent Storage externally and implements information filters as query interfaces. Filtered data are then pipelined into the monitoring service.

2.6. Monitoring service

Thanks to the REST API, the monitoring service works continuously by constantly looking for anomalies within the collected data, focusing on new data, and performing analyses on past data. Detecting anomalies has multiple purposes; first, spotting clues for possible attacks by malicious users or, in the event of an attack, trying to trace the causes. In the longer term, detecting anomalies and errors helps find programming bugs or unwanted system behaviors that often turn into vulnerabilities. Furthermore, it provides datasets for predicting system

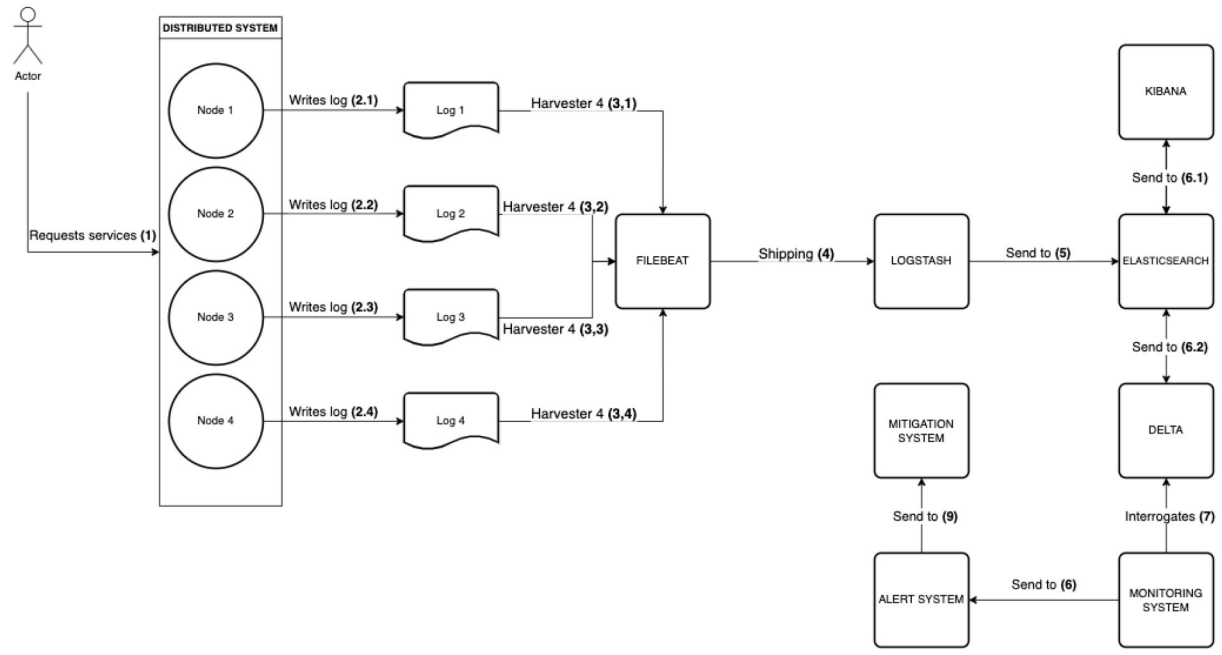

FIGURE 2
Data flow diagram.

behavior and immediately identifying anomalies if the expected behavior does not occur. The design of the monitoring service is very dynamic and strictly depends on the targets to be sentineled; however, it always sends alert messages concerning observed anomalies to the Mitigation Services.

2.7. Mitigation services

Once the monitoring service identifies an anomaly, an alert is sent to the Mitigation Services containing the data about the anomaly identified, such as its type and how it occurred. Following the alert, the Mitigation Services have the task of intervening in the system to control the anomaly. Implementing the Mitigation Services is strictly contextual to the system being sentineled as they rely on system-dependent mitigations, which can be more or less invasive, an aspect to be defined and decided at development time; e.g., operating directly on the nodes that generated the anomaly to mitigate a particular vulnerability or attack in progress or forwarding an additional alert message to the system administrators to verify the nature of the anomaly. Further design options include a single central mitigation system to deal with all mitigation operations or independent services for each macro-type of mitigation required. Each choice of mitigation is characterized by its specific way of resource management and implementation of executable processes. For instance, if there is a single mitigation service, the procedures could be carried out sequentially so as not to create interferences; if there are more mitigation services, the operations could be carried out in parallel, ensuring faster response to anomalies.

2.8. Analysis flow

The analysis flow is described in Figure 2. When a user requests services (1), the system nodes operate according to the codified business logic. While executing operations, the nodes also write logs (2), which are produced separately. The *Filebeat* tool listens, by using *harvesters*, to the streams of logs being written (3) and ships them to *Logstash*, thus attaining their aggregation and centralization (4). *Logstash* applies the data model to the received streams and indexes them on *Elasticsearch* (5). *Kibana* accesses the indexes to make data available to system administrators (6.1). *DELTA* reads the indexes and applies *NLP* techniques on the logs, classifying them and also computing sentiment polarities and extracting keywords (6.2). *DELTA* provides. Additionally, *DELTA* provides APIs to let monitoring systems make queries and analyze in real time whatever may be affecting the system (7); if needed, they will launch alerts (8) to trigger mitigation actions (9).

3. DELTA implementation

The developed *DELTA* system, with components as in Figure 3, is divided into two macro sections, the first relating to the collection of logs through the use of the *Elastic* components and the second consisting of the textual analysis systems provided by the *DELTA* tool. The following sections describe the methodologies used to implement the system: Section 3.1 explains the use of the *Elastic* stack and the log flow within the system; Section 3.2 is instead dedicated to the illustration of the developed tool and how the log analysis and their re-elaboration takes place.

3.1. Elastic stack

The acquisition of logs produced by distributed services and systems is managed through the combined use of several *Elastic* components, namely *Filebeat*, *Logstash*, *Elasticsearch*, and *Kibana*, all assigned to preparing logs for analysis. Specifically, *Filebeat* is responsible for real-time log collection, launching probes on each distributed component output stream channel, such as *stdout* and *stderr*, as well as on the log files written by the system components, including docker logs. These probes, called harvesters, send data collected from the sources defined in the *filbeat.yml* configuration file to *Logstash* that aggregates and stores them in the *Elasticsearch* database. The logs, normalized by using data models, can thus be accessed through a single or a few sources, enabling efficient log analysis and addressing the problem due to many log files. After these steps, Elasticsearch makes indexing of the entered data available for access via queries and REST APIs. Elasticsearch indexed data are suitable, additionally, to be accessed by *Kibana*, which uses them for real-time consulting through a dashboard composed of customizable cs and lists.

3.2. DELTA analyser

DELTA is aimed at processing the logs in the Elasticsearch database to extract relevant information using Natural Language Processing techniques to facilitate data monitoring and analysis

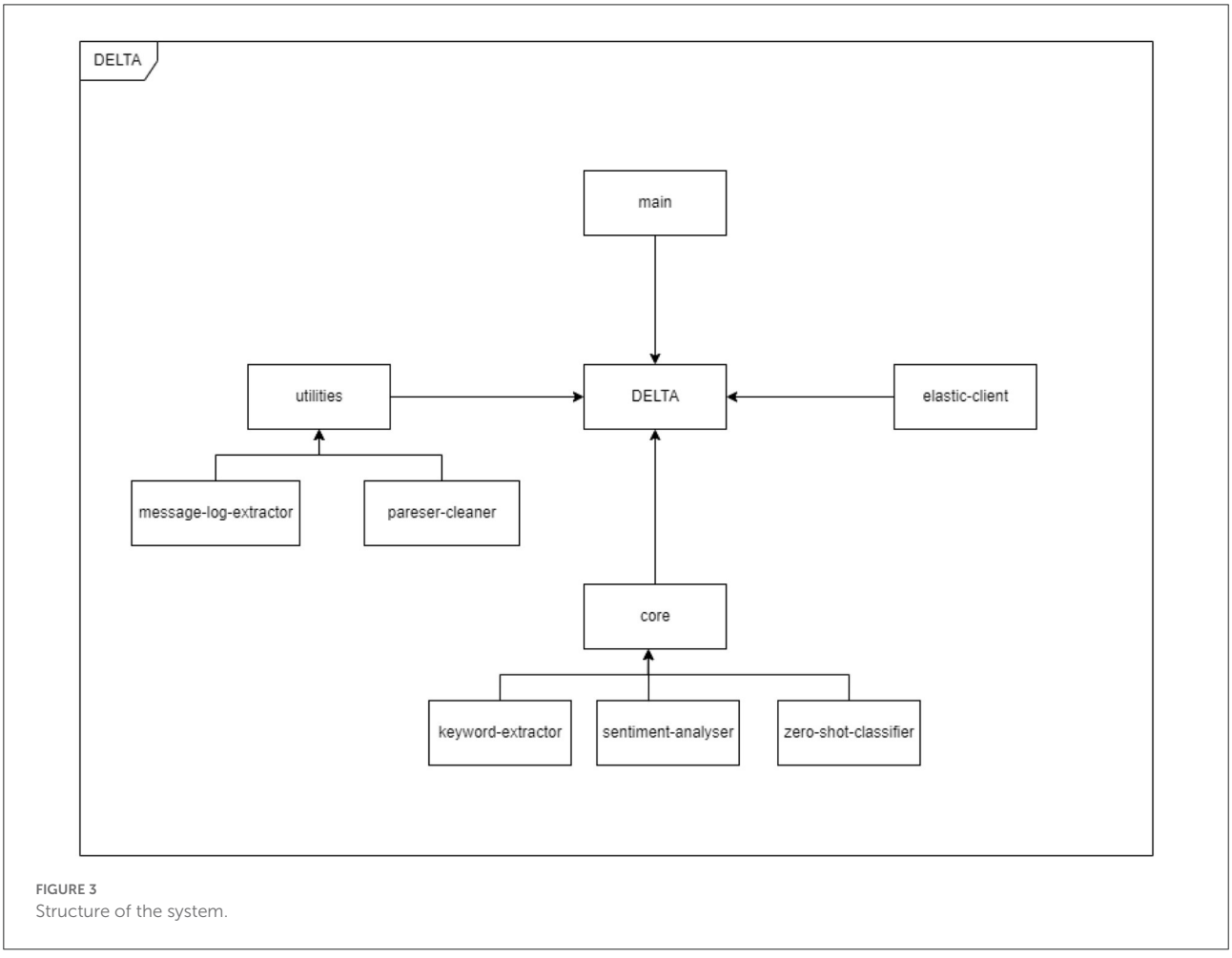

FIGURE 3
Structure of the system.

operations. This occurs through a continuous activity of detection of relevant patterns such as the presence of IP addresses, as well as of processing on the following three dimensions (Figure 4):

- Keywords extraction
- Category classification
- Sentiment analysis

The *DELTA* analytical components and the structure of their relationship are illustrated in Figure 5, which can itself be considered a specific instance of the Analyzer Core as part of the overall architecture depicted in Figure 1.

3.3. Keywords extraction

Keyword extraction picks out words with the highest informative impact on the text, thus making it possible to conduct statistical analysis or associate keywords with the triggering of events across vastly varying contexts. Moreover, keywords can be exploited to boost monitoring effectiveness to access the logs by filtering their content.

Many methods, techniques, and algorithms extract text keywords or key phrases (e.g., TF-IDF, Rake, YAKE!). Since *DELTA* was designed for a generic context, we cannot make domain-oriented predictions about the input to be analyzed. Vice versa, given the specificity that characterizes some application contexts, extraction models may be based on statistical concepts that ignore domain-related keywords. When dealing with a distributed system that uses specific terms to identify its components (for example, the word *Tangle* in the context of the distributed ledger *IOTA*), such elements would be ignored by most models based on word frequency or dictionaries. These considerations led to the implementation of both a keyword extraction technique based on the semantic similarity of the words and a method based on the statistical properties of the texts. Therefore, for keyword extraction, we relied on two models, namely *KeyBERT* and *YAKE!*.

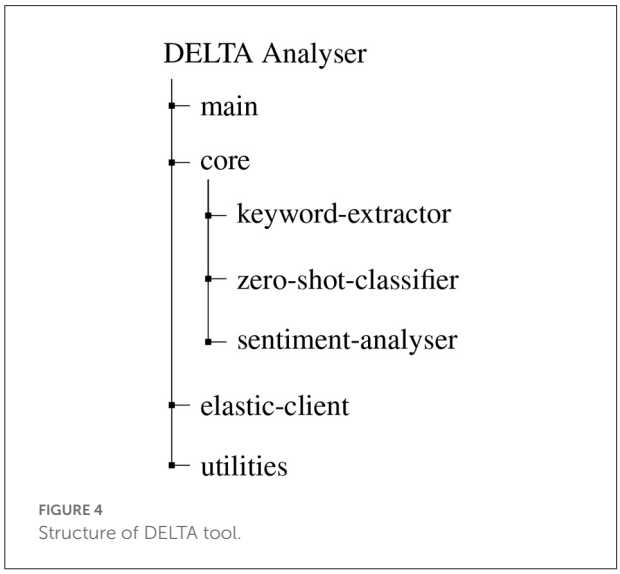

FIGURE 4
Structure of DELTA tool.

3.3.1. KeyBERT

KeyBERT is based on *BERT* (Bidirectional Encoder Representations from Transformers) (Johri et al., 2021), published by Google in 2018, a model that uses a *transformer* (Wolf et al., 2019) architecture to map words, phrases and text into number vectors that capture their meaning. *BERT* is a bidirectional model, which makes it able to interpret a word based on the context of the sentence, regardless of whether the relevant information is left or right, unlike left-to-right architectures, which look only at the words preceding the one being processed (Devlin et al., 2018). Furthermore, *BERT* does not resort to recursive levels unlike *LSTM*-based technologies (Hochreiter and Schmidhuber, 1997), but instead exploits the *Self-Attention* (Cui et al., 2019) mechanism. The *KeyBERT* system can be implemented with different *transformer* models, with the basic model *all-MiniLM-L6-v2* having limited needs for computational resources with a trade-off in precision levels. While, according to the official documentation, the highest quality model turns out to be *all-mpnet-base-v2*, we have chosen to use the less powerful *paraphrase-albert-large* model, as log texts are generally compact, so this model achieves excellent accuracy with lower resource consumption.

3.3.2. YAKE!

YAKE! or *Yake!* or *Yake*, is an automatic keyword extraction algorithm that stands out above all for its simplicity and an excellent balance between computational resource requirements and the quality of the analytics. It is an unsupervised algorithm based on statistical textual characteristics extracted from individual documents. Therefore, it does not need to be trained on a particular set of documents or depend on external dictionaries and *corpora*, nor has limitations regarding text sizes, languages, or domains. It also does not use *Part of Speech Tagging* (Voutilainen, 2003), which makes it language-independent, except for using different but static stopword lists for each language. This makes it easy to apply to languages other than English, particularly low-diffusion languages for which open-source language processing tools may be underperforming.

3.3.3. KeyBERT compared to YAKE!

KeyBERT and *YAKE!* thus provide alternative models for keyword extraction. In testing the two algorithms on logs, we

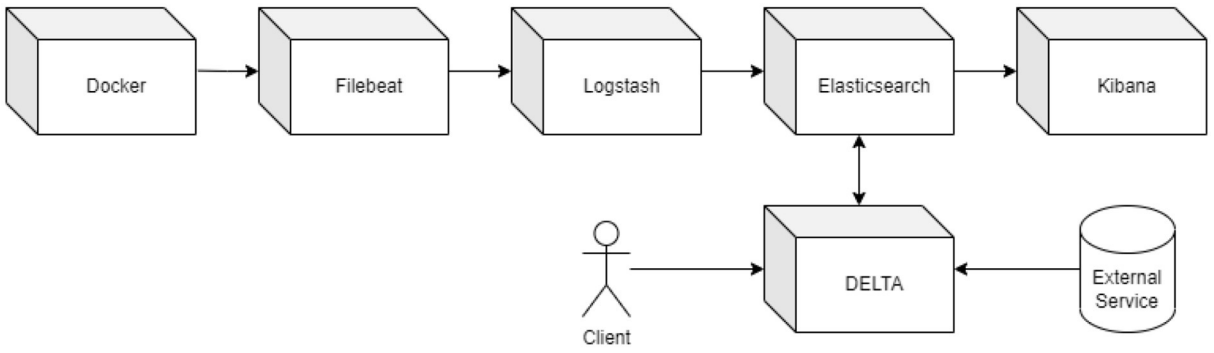

FIGURE 5
Structure of the DELTA implementation.

found that both offer high accuracy for short texts. Nevertheless, the model suggested by default is *KeyBERT*, in virtue of its higher accuracy for longer texts. On the other hand, *KeyBERT* turns out to be significantly more onerous in terms of performance and waiting time. Therefore, *DELTA* provides both approaches to let users choose the most suitable one for the analysis context.

3.4. Log classification

Classifying logs according to labels (i.e., classification categories) can help analyze and monitor distributed systems. First, it is thus possible to get an idea of the frequency with which logs that share the same label occur to facilitate the identification of related problems. Furthermore, through the analysis of the logs of the same category, it is possible to identify the presence of specific patterns usable to verify system functioning or detect anomalies attributable to errors or tampering attempts. Finally, labels offer a way to access content in addition to keyword-based querying. The classification approach used in *DELTA* is a hybrid that combines machine learning with rules, being a *Zero-Shot* (Gera et al., 2022) classifier. As illustrated in Yin et al. (2019), this methodology classifies text, documents, or sentences without resorting to any previously tagged data by using a natural language processing model, pre-trained in an environment or domain that may be completely different from the application domain, hence making it possible to classify texts from heterogeneous contexts. It provides a probabilistic value of whether the text belongs to a label by taking a text and a list of possible labels as input. Thus, through a threshold, the text is labeled according to the categories to which it belongs. As used in *DELTA*, *Zero-Shot* provides multiple labels to be assigned to a single log with different probabilistic scores by exploiting the model output as an independent probabilistic value for each supplied label. It follows a more detailed view of the system behavior and the possibility of monitoring logs in a more specific and targeted way. There are five preset tags used within *DELTA*: *Security*, *Connection*, *Communication*, *Transaction* and finally *Operation*, changeable according to the context of use. The currently adopted model is the one provided by *Facebook*: *"facebook/bart-large-mnli"* (Adaszewski et al., 2021), but this too can be changed according to needs and preferences.

3.5. Sentiment analysis

In addition to keyword extraction and classification analysis, log sentiment analysis is also performed. Sentiment analysis consists of language processing and analysis aimed at identifying the subjective value of the text, with the primary goal to determine the polarity of a document, i.e., to classify it according to the degree of positivity, negativity, or neutrality of the concepts expressed. As surprising as it may seem, logs carry sentiment that can usefully shed light on what is happening within the monitored system. A monitoring system based on *DELTA* will indeed benefit from the identification of logs loaded with negative sentiment indicative of errors or malfunctions to detect anomalies and errors. For the extraction of sentiment from the logs, the library *VADER-lexicon general* is used, which, born as a sentiment analysis library aimed at social media and customer feedback, has valuable features that make it an excellent analyzer for short texts and hence for logs too. Once a given text has been analyzed, *VADER* responds with a polarity value called *compound* that indicates the degree of positivity or negativity of the sentiment of the analyzed text. The *compound* is later processed to define a sentiment evaluation label through thresholds that were tuned and set according to empirical evidence. Within *DELTA*, it is also possible either to modify the values of the thresholds or to add new ones to refine positivity or negativity levels.

3.6. Additional log processing

Further processing was carried out to bring intrinsically relevant information to the fore. First of all, elements deemed irrelevant for the analysis were removed. We also worked on the *Logstash* component responsible for collecting data and creating a very detailed structure containing all the possible information directly observable from the log generation sources. This structure has been stripped down and simplified as far as possible to facilitate future analyses and speed up information sharing. Furthermore, regarding security aspects and the control of the use of the system, the extraction of IP addresses and connection ports, if present in the logs, was carried out. Finally, the information relating to the log output standard was also extracted, thus making it possible for the aggregation filters to operate in a simplified manner based on the type of log produced by the system's Listing 1 in Appendix.

4. Application to Hyperledger Fabric

We briefly describe a *DELTA* application to Hyperledger Fabric (Androulaki et al., 2018; Dalla Palma et al., 2021), the most adopted platform for building private distributed ledgers. The challenges presented by blockchain and DL management were, in fact, the initial motivation for *DELTA*, even if its construction was then generalized to all systems that can be containerized and distributed through technologies such as Docker and Kubernetes.

Like any distributed ledger, Fabrics goals include ensuring a secure environment. However, it has known vulnerabilities (Andola et al., 2019; Dabholkar and Saraswat, 2019; Paulsen, 2021) that provide attack points for malicious users. While these can be mitigated, there is a lack of a monitoring system that can detect potential attacks and act promptly. This problem can be addressed by using *DELTA* to analyze the logs produced by the system during the attack phase. Log checking is performed to identify attack patterns, and then these are used to develop a monitoring system to spot and mitigate threats in real time. Since Fabric blockchains are distributable using the Docker engine, *DELTA* is a close fit for log analysis of the entire Fabric network. The monitoring system is based on the logs generated by the Docker containers and then analyzed and processed by *DELTA* in real-time. *DELTA* provides REST APIs, used to make queries to filter data written in logs and bring applicants such as monitoring systems the data related to attack detection; in their turn, monitoring systems will send alerts to the mitigation services to contain the attack if certain conditions associated with the attack patterns are met. To this aim, several additional valuable data are extractable, such as name, unique ID image, execution status, and installed volumes of Docker containers. To guarantee generality, the detection logic can be, of course, customizable. Moreover, *DELTA* provides keywords, sentiment, and the types of the log, which can be Security, Connection, Communication, Transaction, and Operation. All the information extracted by *DELTA* is available on Kibana, which provides customizable dashboards and charts to make visible, what is happening in the system, and integrate, the NLP information extracted by *DELTA*, namely, sentiment, most relevant keywords, and classification.

DELTA has been successfully used to detect network-based attacks on Hyperledger Fabric (Pilla et al., 2022), especially against Distributed Denial of Service (Lau et al., 2000), which exploits some vulnerabilities of Fabric, such as the relatively lower level of decentralization, compared to other blockchains, resulting from design choices like the use of a centralized Ordering Service for transaction management, and the exposition of *on-chain* services provided by the Blockchain as a Service (BaaS), which enables clients to requests transactions and receive results, elaborated by *on-chain* smart contracts. Analyzing distributed systems logs, *DELTA* can classify and detect attacks, giving information to clients who call DELTA APIs. Hence, all the information provided by *DELTA* can be used by monitoring systems. The monitoring system implemented in our prototype aims to manage the prevention of network attacks and works by sending queries to *DELTA*. Whenever it detects patterns related to attacks, relying on *DELTA* to identify possible dangers, and once the anomalous behavior is confirmed, it proceeds to activate a mitigation service that sends warning messages via webhook to all configured addresses on the basis of the detections carried out.

A real test case of *DELTA* monitoring DDOS attacks on the Ordering Service highlighted its effectiveness over traditional methods. The typical way the attack scenario is realized is by continuously requesting access to clog the information flow of the Ordering Service until it gets completely bogged down (Andola et al., 2019; Dabholkar and Saraswat, 2019). The traditional way to mitigate this attack is complex and challenging, involving, as it does, constantly monitoring performance and resource usage metrics, such as throughput and transaction latency, for early detection of compromised availability. With the proposed log analysis strategy, we can monitor the attack scenario simply by analyzing the logs in real time thanks to *DELTA*, following the flow illustrated in Figure 2. An alert will be launched as soon as the system produces the logs associated with the Ordering Peer requests. Subsequently, the attack can be mitigated based on the customizable policies of the user organizations in charge of system governance. For example, one possible approach may be to close the channel of request-flooding peers.

5. Related work

We have already mentioned in the Introduction the tools currently used for the security of cloud-based systems (Kufel,

2016) and highlighted their limitations concerning the auditing requirements of blockchains and distributed ledgers. Several contributions (Cucurull and Puiggali, 2016; Ahmad et al., 2018; Li et al., 2020; Ali et al., 2022) should be mentioned that are orthogonal to ours, in that rather than the analytics on logs generated in distributed environments such as blockchains and distributed ledgers, they are aimed at using blockchains for secure log management. The approach described in Regueiro et al. (2021) is in line with those above. However, it comes close to ours in the use of Elastic Search and Kibana for log indexing and visualization, but then, while we are platform-independent, is bound in its implementation to Quorum, the permissioned blockchain used for log management, and does not provide for advanced analytics capabilities such as log sentiment analysis and classification.

In a completely different category belong commercial systems based on artificial intelligence dedicated to defending organizations from various cyber-attacks, ranging from denial-of-service to ransomware. The most well-known providers of this type of solution are companies such as Darktrace (2022) and Vectra (2022). On the one hand, these approaches have common ground with ours due to their extensive use of advanced analytics techniques based on artificial intelligence and machine learning; on the other hand, they diverge in how they build defenses due to the very different organizational contexts they target. These tools focus on individual organizations, even though they are committed to dealing with attacks that originate in a highly distributed and diversified external world and can come from various fronts and directions. Thus they function by empowering and automating with artificial intelligence and machine learning a knowledge base built from the experience of these organizations in defending against external attacks. By contrast, our framework targets inter-organizational vulnerabilities that arise within multi-organization ecosystems that rely on and evolve on the infrastructure provided by distributed systems such as blockchains and distributed ledgers. The future will increasingly see the emergence of business ecosystems characterized by multi-company governance in support of complex production interdependencies in the context of trends such as the development of global supply chains and frameworks such as Industry 4.0. All this will give more and more space for systemic approaches like ours, aimed at listening to what the system tells us and complementing and extending those monitoring the individual organizations that make it up.

6. Conclusion and future work

The exponential growth of distributed systems in recent years, including blockchains and distributed ledgers, has led to greater attention to these systems' security and analysis issues. In particular, the security of the information present within distributed systems is paramount because these systems are increasingly deployed in contexts characterized by sensitive information. For this purpose, we have defined a general architecture for log auditing in distributed systems. On its basis, we have designed and implemented the *DELTA* tool that collects and stores logs generated by the services that make up the system through some of the components provided by the *Elastic* search ecosystem for data analytics. Once the logs are suitably processed to simplify their analysis, their text content goes through Natural Language Processing to extract keywords and sentiment and is classified according to relevant categories. Keywords enable effective log search, and their extraction can be done according to needs by choosing between *KeyBERT*, more precise, and *YAKE!*, faster and lighter. Sentiment analysis is performed through the *VADER* algorithm to measure the degree of text sentiment, where a significant degree of negativity warns about the need to carry out thorough checks on what is happening to the system. Classified logs can be set based on the characteristics of the execution system to access them according to classification. The purpose of these analytical capabilities is to effectively provide the information extracted from the logs to external monitoring processes, which can thus carry out specific and detailed analyses based on the problems at hand and mitigate them. To this end *DELTA* was made customizable and is interfaceable with other platforms through REST APIs to query the system and suitably filter content.

The *DELTA* tool provides an initial log auditing approach specific to distributed systems with a focus on blockchain and *DLT* platforms. However, it is limited to the *Docker engine*. Although widely used and suitable for distributed systems, *Docker* does not scale effectively to very large systems. Consequently, the next step will be to integrate *DELTA* with *Kubernetes* (Mondal et al., 2022), an open-source platform, initially developed by *Google*, for managing workloads and orchestrating containerized services, which simplifies both system configuration and automation of service delivery practices in very large systems.

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found here: https://www.kaggle.com/datasets/zappaboy/hyperledger-fabric-logs.

Author contributions

PD designed and implemented the *DELTA* tool. RP managed and coordinated the development of the described framework and the writing of this article. FS identified the testing cases for *DELTA* in Hyperledger Fabric and carried out the tests. FZ defined the general architecture and supervised the implementation phases. All authors contributed to the writing of the article.

Funding

The work was partially funded through the Novatellus project financed by the Italian Ministry for Economic Development (MISE) and through the WE_BEST project financed by the Italian Ministry of University and Research (MUR).

Conflict of interest

RP and FZ are founding partners of BB-Smile Srl, a university spin-off jointly promoted by the University of Molise and the University of Rome La Sapienza.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Adaszewski, S., Kuner, P., and Jaeger, R. J. (2021). Automatic pharma news categorization. *arXiv preprint* arXiv:2201.00688.

Ahmad, A., Saad, M., Bassiouni, M. A., and Mohaisen, A. (2018). "Towards blockchain-driven, secure and transparent audit logs," in *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, MobiQuitous, New York City, NY*, eds H. Schulzrinne and P. Li (New York, NY: ACM), 443-448.

Alekseev, A., Megino, F., Klimentov, A., Korchuganova, T., Maendo, T., and Siarhei, P. (2018). Building analytical platform with big data solutions for log files of panda infrastructure. *J. Phys. Conf. Ser.* 1015:032003. doi: 10.1088/1742-6596/1015/3/032003

Ali, A., Khan, A., Ahmed, M., and Jeon, G. (2022). BCALS: blockchain-based secure log management system for cloud computing. *Trans. Emerg. Telecommun. Technol.* 33, e4272. doi: 10.1002/ett.4272

Andola, N., Gogoi, M., Venkatesan, S., Verma, S., et al. (2019a). Vulnerabilities on hyperledger fabric. *Pervasive Mobile Comput.* 59, 101050. doi: 10.1016/j.pmcj.2019.101050

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., et al. (2018). "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 1–15. doi: 10.1145/3190508.3190538

Bavaskar, P., Kemker, O., and Sinha, A. (2020). A survey on: "log analysis with elk stack tool. *SSRN Electron. J.* 6, 2348–1269.

Bottoni, P., Gessa, N., Massa, G., Pareschi, R., Selim, H., and Arcuri, E. (2020). Intelligent smart contracts for innovative supply chain management. *Front. Blockchain* 3, 52. doi: 10.3389/fbloc.2020.535787

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C., and Jatowt, A. (2020). Yake! keyword extraction from single documents using multiple local features. *Inform. Sci.* 509, 257–289. doi: 10.1016/j.ins.2019.09.013

Cangemi, M. P. and Brennan, G. (2019). Blockchain auditing—accelerating the need for automated audits! *EDPACS* 59, 1–11. doi: 10.1080/07366981.2019.1615176

Carlini, F., Carlini, R., Dalla Palma, S., Pareschi, R., and Zappone, F. (2020). The genesy model for a blockchain-based fair ecosystem of genomic data. *Front. Blockchain* 3, 57. doi: 10.3389/fbloc.2020.483227

Crystal-team (2019). *Crypto Auditing: A Compromise Between Enterprises and Regulators*. Available online at: https://medium.com/@CrystalBitfury/crypto-auditing-a-compromise-between-enterprises-and-regulators-4b735f7e7e36

Cucurull, J. and Puiggali, J. (2016). "Distributed immutabilization of secure logs," in *Security and Trust Management-12th International Workshop, STM 2016, Heraklion, Crete, Greece, September 26-27, 2016, Proceedings*, volume 9871 of *Lecture Notes in Computer Science* (Berlin: Springer), 122–137. doi: 10.1007/978-3-319-46598-2_9

Cui, B., Li, Y., Chen, M., and Zhang, Z. (2019). "Fine-tune BERT with sparse self-attention mechanism," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (Hong Kong, China). Association for Computational Linguistics), 3548–3553.

Dabholkar, A. and Saraswat, V. (2019). "Ripping the fabric: attacks and mitigations on hyperledger fabric," in International Conference on Applications and Techniques in Information Security (Berlin: Springer), 300–311. doi: 10.1007/978-981-15-0871-4_24

Dalla Palma, S., Pareschi, R., and Zappone, F. (2021). "What is your distributed (hyper) ledger?," in *2021 IEEE/ACM 4th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*. IEEE, 27–33.

Darktrace (2022). *Dark Trace Cyber AI Analyst: Autonomous Investigations*. Available online at: https://darktrace.com/resources/darktrace-cyber-ai-analyst-autonomous-investigations

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. N. (2018). *Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding.*

Dorri, A., Steger, M., Kanhere, S. S., and Jurdak, R. (2017). Blockchain: a distributed solution to automotive security and privacy. *IEEE Commun. Mag.* 55, 119–125. doi: 10.1109/MCOM.2017.1700879

Fosso Wamba, S., Kala Kamdjoug, J. R., Epie Bawack, R., and Keogh, J. G. (2020). Bitcoin, blockchain and fintech: a systematic review and case studies in the supply chain. *Prod. Plan. Control* 31, 115–142. doi: 10.1080/09537287.2019.1631460

Gera, A., Halfon, A., Shnarch, E., Perlitz, Y., Ein-Dor, L., and Slonim, N. (2022). Zero-shot text classification with self-training. *arXiv [Preprint]*. doi: 10.48550/arXiv.2210.17541

Glaser, B. G. and Strauss, A. L. (2017). *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Milton Park: Routledge.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.* 9, 1735–1780. doi: 10.1162/neco.1997.9.8.1735

Johri, P., Khatri, S. K., Al-Taani, A. T., Sabharwal, M., Suvanov, S., and Kumar, A. (2021). "Natural language processing: History, evolution, application, and future work," in *Proceedings of 3rd International Conference on Computing Informatics and Networks* (Berlin: Springer), 365–375.

Knownsec-404 (2019). *Ethereum Smart Contract Audit Checklist*. Available online at: https://medium.com/@knownsec404team/ethereum-smart-contract-audit-checklist-ba9d1159b901

Kufel, Ł. (2016). Tools for distributed systems monitoring. *Found. Comput. Decis. Sci.* 41, 237–260. doi: 10.1515/fcds-2016-0014

Lau, F., Rubin, S. H., Smith, M. H., and Trajkovic, L. (2000). "Distributed denial of service attacks," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics: Cybernetics Evolving to Systems, Humans, Organizations, and their Complex Interactions, Sheraton Music City Hotel* (Nashville, TN). IEEE, 2275–2280.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*. doi: 10.18653/v1/2020.acl-main.703

Li, J., Wu, J., Jiang, G., and Srikanthan, T. (2020). Blockchain-based public auditing for big data in cloud storage. *Inf. Process. Manag.* 57, 102382. doi: 10.1016/j.ipm.2020.102382

Mondal, S. K., Pan, R., Kabir, H. M. D., Tian, T., and Dai, H. (2022). Kubernetes in IT administration and serverless computing: an empirical study and research challenges. *J. Supercomput.* 78, *2937–2987*. doi: 10.1007/s11227-021-03982-3

Motta, G. A., Tekinerdogan, B., and Athanasiadis, I. N. (2020). Blockchain applications in the agri-food domain: the first wave. *Front. Blockchain* 3, 6. doi: 10.3389/fbloc.2020.00006

Paulsen, C. (2021). *Revisiting Smart Contract Vulnerabilities in Hyperledger Fabric*. Available online at: https://cse3000-research-project.github.io/static/d23cb4583e6d97a1e509eafda859c424/poster.pdf

Pilla, P. D., Pareschi, R., Salzano, F., and Zappone, F. (2022). "Hyperledger fabric attacks mitigation (extended abstact)," in *FOCODILE 2022—3rd International Workshop on Foundations of Consensus and Distributed Ledgers*.

Regueiro, C., Seco, I., Gutiérrez-Agüero, I., Urquizu, B., and Mansell, J. (2021). A blockchain-based audit trail mechanism: Design and implementation. *Algorithms* 14, 341. doi: 10.3390/a14120341

Reis, D., Piedade, B., Correia, F. F., Dias, J. P., and Aguiar, A. (2022). Developing docker and docker-compose specifications: A developers' survey. *IEEE Access* 10, 2318–2329. doi: 10.1109/ACCESS.2021.3137671

Stol, K., Ralph, P., and Fitzgerald, B. (2016). "Grounded theory in software engineering research: a critical review and guidelines," in *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016*, eds L. K. Dillon, W. Visser, and L. A. Williams (Austin, TX). ACM, 120–131.

Vectra (2022). *The AI Behind Vectra AI*. Available online at: https://content.vectra.ai/hubfs/downloadable-assets/WhitePaper_2022_ai_behind_vectra_ai.pdf

Voutilainen, A. (2003). "Part-of-speech tagging," *The Oxford Handbook of Computational Linguistics* (Oxford: OUP), 219–232.

Wang, Q., and Su, M. (2020). Integrating blockchain technology into the energy sector' from theory of blockchain to research and application of energy blockchain. *Comput. Sci. Rev.* 37, 100275. doi: 10.1016/j.cosrev.2020.100275

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., et al. (2019). Huggingface's transformers: state-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*. doi: 10.18653/v1/2020.emnlp-demos.6

Yin, W., Hay, J., and Roth, D. (2019b). "Benchmarking zero-shot text classification: datasets, evaluation and entailment approach," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, eds K. Inui, J. Jiang, V. Ng, and X. Wan (Hong Kong, China). Association for Computational Linguistics, 3912–3921.

Zheng, Z., Xie, S., Dai, H., Chen, X., and Wang, H. (2017). "An overview of blockchain technology: architecture, consensus, and future trends," in *2017 IEEE International Congress on Big Data (BigData congress)*. IEEE, 557–564. doi: 10.1109/BigDataCongress.2017.85

10.3389/fcomp.2022.1020946

Appendix

Structure of Processed Logs

```
{
   "_index":"data_parsed",
   "_type":"_doc",
   "_id":"HkEy0n8BYv2pyDWhsjXR",
   "_score":1.0,
   "_source":{
      "@timestamp":"2022-03-20T12:48:09.287Z",
      "type":"ERROR",
      "message":"[31m2022-03-20 12:48:09.287 UTC 007d ERRO [
           0m [core.comm] [31;1mServerHandshake [0m ->
           Server TLS handshake failed in 607.238772ms with
            error server=Orderer remoteaddress=167.94.138.4
           6:45236",
      "sentiment":"very negative",
      "id_log":"MQhep38BYv2pyDWh9xpE",
      "container":{
         "id":"0c06a22e0a5f43b7d2ef9b6bfbfa227ae828a3fd2be0e
              1ab44e3f46926106640",
         "name":"orderer.example.com",
         "image":{
            "name":"hyperledger/fabric-orderer:2.4.2"
         }
      },
      "keywords":[
         [
            "1mserverhandshake",
            "remoteaddress",
            "failed"
         ]
      ],
      "classification_labels":[
         [
            "Communication",
            "Security",
            "Connection"
         ]
      ],
      "ip":[
         "167.94.138.46:45236"
      ],
      "name_image_doc":{
         "name":"hyperledger/fabric-orderer:2.4.2"
      },
      "stream":"stderr"
   }
}
```

Listing 1 Simplified structure of processed logs used on Hyperledger Fabric network.