DELTA - Distributed Elastic Log Text Analyser

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*

Abstract
Distributed systems have become ubiquitous in recent years, with those based on distributed ledger technology (DLT), such as blockchains, gaining more and more weight. Indeed, DLT ensures strong data integrity thanks to complex cryptographic protocols and high distribution. That said, even the most powerful systems will never be perfect, and, in fact, the larger they get, the more exposed they become to threats. For traditional systems, log auditing effectively addresses the problem and makes it possible to analyze the use of applications. However, DLT systems still lack a wide range of log analyzers due to the particularities of their distribution. To help remedy this weakness, we propose here a generic auditing system called *DELTA* (for Distributed Elastic Log Text Analyzer). By coupling Natural Language Processing with the *Docker Engine* of the *Filebeat, Logstash stack, Elasticsearch* and the visual tool *Kibana*, *DELTA* tracks, analyzes and classifies logs generated by DLT systems. Additionally, it enables real-time monitoring thanks to visual analysis and querying of structured data. *DELTA* is the first auditing system applicable to blockchains that can be integrated with the *Docker Engine*. In addition to describing its general principles and specific components, we illustrate its application to Hyperledger Fabric, the most popular of the platforms for building private blockchains.

Keywords
Distributed Ledger Technology, Log Analysis, Cybersecurity, Natural Language Processing, Blockchain, NLP, DLT .

1. Introduction

Distributed systems have been spreading rapidly in recent years [30], 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. [2, 27, 21, 8, 5, 15, 13]. 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

DLT 2022: 4th Distributed Ledger Technology Workshop, June 20, 2022, Rome, Italy
*Corresponding author.
†These authors contributed equally.
p.dipilla@studenti.unimol.it (P. D. Pilla); remo.pareschi@unimol.it (R. Pareschi); f.salzano1@studenti.unimol.it (F. Salzano); federico.zappone@unimol.it (F. Zappone)
http://docenti.unimol.it/index.php?u=remo.pareschi (R. Pareschi); https://github.com/ZappaBoy (F. Zappone)
0000-0001-6455-6575 (F. Zappone)

for the development of decentralized applications (*dApps*). Data security is a crucial issue for information systems in general, so, 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, system 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 the behavior of IT systems under both 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. 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 [18] 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 [7]. We started from these premises in carrying out 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* and the *stack ELK* (*Elasticsearch, Logstash Kibana*) integrated via *Filebeat*.

For this purpose, we make use of 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 is not limited to traceability in that the textual part of the traced logs is subjected to analysis through Natural Language Processing (NLP). NLP is used to perform, upon the text within logs produced by the *Docker containers*, three types of analysis, namely: 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!* [6]. 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 [20]. 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. In 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 methodology underlying *Delta* and the components used for its implementation; Section 3 describes the application of DELTA to Hyperledger Fabric, the most popular platform for private blockchains; Section 4 describes future work; Section 5 concludes the paper.

2. Methodology

The developed system is mainly divided into two macro sections, the first relating to the collection of logs through the use of the *Elastic* components and the second part consisting of the textual analysis systems provided by the *DELTA* tool. The following sections describe the methodologies used to implement the system: Section 2.1 describes the use of the *Elastic* stack and the log flow within the system, Section 2.2 is instead dedicated to the illustration of the developed tool and how the log analysis and their re-elaboration takes place.

2.1. Elastic stack

The acquisition of logs produced by distributed services and systems was managed through the combined use of several *Elastic* components. In fact, the stack used includes 4 such components, namely *Filebeat*, *Logstash*, *Elasticsearch* and *Kibana*. The *Filebeat* component has been added on top of the standard *ELK* stack as it supports log extraction from highly heterogeneous contexts and therefore is perfectly suited to distributed environments.

These components fit together to obtain, manage, index, and view the generated logs automatically and instantly. The logs come from the containers where the distributed services are located and are first extracted through *Filebeat*, which reads the data directly by connecting to the log files managed by the *Docker engine*. Then *Filebeat* passes the logs to *Logstash* which aggregates them and places the raw information inside *Elasticsearch* creating a special index for archiving. Finally, *Elasticsearch* makes indexing of the entered data available for access via *Kibana* which provides the visual analysis of both the aggregate data entered through the collection process and of the analyzed data (Figure 1).

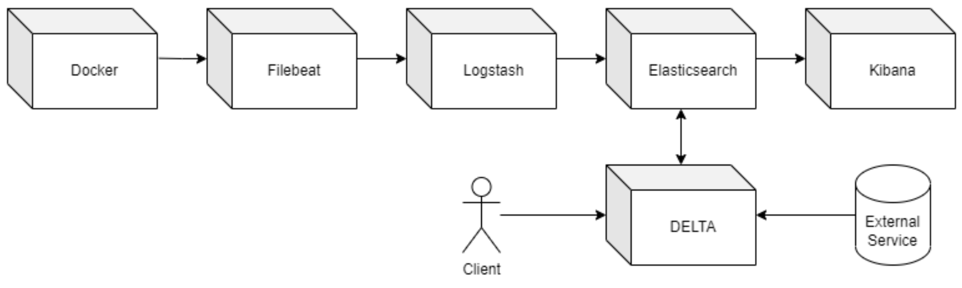

Figure 1: Structure of the system

2.2. 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 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 2):

- **Keywords Extraction**
- **Category Classification**
- **Sentiment Analysis**

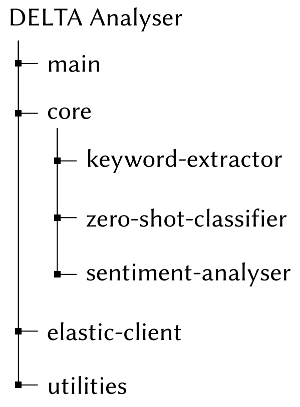

Figure 2: Structure of DELTA tool

2.3. 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 keywords or key phrases from texts (e.g., TF-IDF, Rake, YAKE!). Since *DELTA* was designed for a generic context, we cannot make predictions about the input to be analyzed. Vice versa, given the specificity that characterizes some application contexts, there may be extraction models based on statistical concepts that ignore specific 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 the most models based on word frequency or dictionaries. These considerations led to implementing 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!*.

2.3.1. KeyBERT

KeyBERT is based on *BERT* (Bidirectional Encoder Representations from Transformers) [17], published by Google in 2018, a model that uses a *transformer* [28] 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 [12]. Furthermore, *BERT* does not resort to recursive levels unlike *LSTM*-based technologies [16], but instead exploits the *Self-Attention* [9] 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.

2.3.2. 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 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, nor does it depend on external dictionaries and *corpora*, nor has limitations as regards text sizes, languages, or domains. It also does not use *Part of Speech Tagging* [26], which makes it language-independent, except for the use of 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.

2.3.3. KeyBERT compared to YAKE!

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

2.4. 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, which can then be used to verify the system's correct 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* [24] classifier. As illustrated in [29], 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. This makes 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. The way *Zero-Shot* is used in *Delta* provides for multiple labels to be assigned to a single log with different probabilistic scores by exploiting the fact that the model output is an independent probabilistic value for each supplied label. This gives a more detailed view of the system behavior and the consequent 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*. However, these can be changed according to the context of use. The currently adopted model is the one provided by *Facebook*: *'facebook/bart-large-mnli'* [1], but this too can be changed according to needs and preferences.

2.5. 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 subjectivity 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 going on within the monitored system. In order to detect anomalies and errors, a monitoring system based on *DELTA* will indeed benefit from the identification of logs loaded with negative sentiment, indicative of errors or malfunctions. 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 levels of positivity and/or negativity.

2.6. Additional Log Processing

In addition to the textual analysis of the logs, 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,

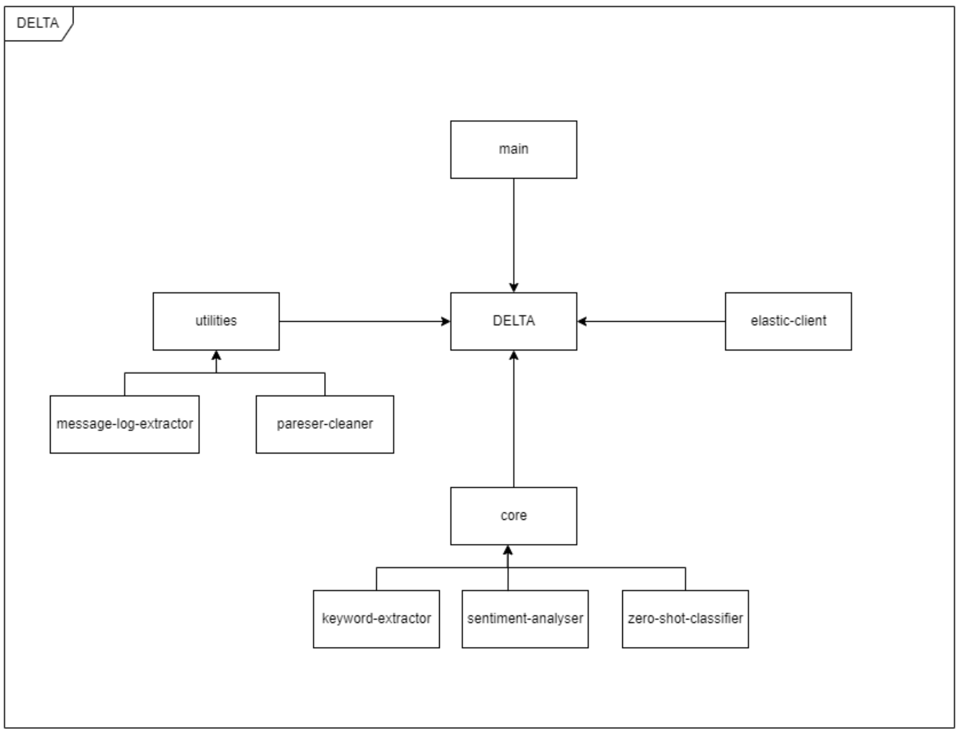

Figure 3: Structure of the system

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 systems (Listing 1).

3. Application to Hyperledger Fabric

We briefly describe a *DELTA* application to Hyperledger Fabric [4, 11], 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, Fabric's goals include ensuring a secure environment. However, it has known vulnerabilities [10, 22, 3] 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. Queries can filter logs and obtain only the relevant ones for attack detection. To this aim, several additional valuable data are extractable, such as name, unique ID image, execution status, and installed volumes of Docker containers. Moreover, *DELTA* provides keywords, sentiment, and the types of the log, which can be Security, Connection, Communication, Transaction, and Operation.

The attacks on Fabric fall into two broad categories, depending on whether they are about the network rather than the execution of smart contracts [23]. *DELTA* is particularly effective at dealing with the first ones. They are essentially variants of well-known attacks in distributed systems, e.g., Distributed Denial of Service[19], Sybil[14] and Eclipse attacks [25], which exploit some specificities 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. A monitoring system aimed at network attacks and consisting of three microservices was consequently implemented, namely *i)* a service that relies on *DELTA* to detect patterns of potential danger, once anomalous behavior is confirmed, sends a warning to all configured addresses, *ii)* a service that takes care of sending warning messages via webhook based on the detections made as in *i)*, *iii)* a dedicated mitigation service.

4. Future Work

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*, 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.

5. Conclusion

The exponential growth of distributed systems in recent years, mainly due to the advent of Distributed Ledger Technology and, in particular, blockchains, 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 being increasingly deployed into contexts characterized by sensitive information. For this purpose, we have designed and implemented the *DELTA* tool that collects and stores logs generated by the services that make up the system through the use of some of the components provided by the *Elastic* search ecosystem for data analytics. Then the logs are suitably processed to simplify their analysis. Finally, 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. Logs are classified in categories that 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 consequently 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.

References

[1] Stanislaw Adaszewski, Pascal Kuner, and Ralf J Jaeger. Automatic pharma news categorization. *arXiv preprint arXiv:2201.00688*, 2021.

[2] Shiroq Al-Megren, Shada Alsalamah, Lina Altoaimy, Hessah Alsalamah, Leili Soltanisehat, Emad Almutairi, et al. Blockchain use cases in digital sectors: A review of the literature. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1417–1424. IEEE, 2018.

[3] Nitish Andola, Manas Gogoi, S Venkatesan, Shekhar Verma, et al. Vulnerabilities on hyperledger fabric. *Pervasive and Mobile Computing*, 59:101050, 2019.

[4] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[5] Paolo Bottoni, Nicola Gessa, Gilda Massa, Remo Pareschi, Hesham Selim, and Enrico Arcuri. Intelligent smart contracts for innovative supply chain management. *Frontiers in Blockchain*, 3:52, 2020.

[6] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289, 2020.

[7] Michael P Cangemi and Gerard Brennan. Blockchain auditing–accelerating the need for automated audits! *EDPACS*, 59(4):1–11, 2019.

[8] Federico Carlini, Roberto Carlini, Stefano Dalla Palma, Remo Pareschi, and Federico Zappone. The genesy model for a blockchain-based fair ecosystem of genomic data. *Frontiers in Blockchain*, 3:57, 2020.

[9] Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 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)*, pages 3548–3553, Hong Kong, China, November 2019. Association for Computational Linguistics.

[10] Ahaan Dabholkar and Vishal Saraswat. Ripping the fabric: Attacks and mitigations

on hyperledger fabric. In *International Conference on Applications and Techniques in Information Security*, pages 300–311. Springer, 2019.

[11] Stefano Dalla Palma, Remo Pareschi, and Federico Zappone. What is your distributed (hyper) ledger? In *2021 IEEE/ACM 4th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pages 27–33. IEEE, 2021.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[13] Ali Dorri, Marco Steger, Salil S Kanhere, and Raja Jurdak. Blockchain: A distributed solution to automotive security and privacy. *IEEE Communications Magazine*, 55(12):119–125, 2017.

[14] John R. Douceur. The sybil attack. In Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron, editors, *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer, 2002.

[15] Samuel Fosso Wamba, Jean Robert Kala Kamdjoug, Ransome Epie Bawack, and John G Keogh. Bitcoin, blockchain and fintech: a systematic review and case studies in the supply chain. *Production Planning & Control*, 31(2-3):115–142, 2020.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[17] Prashant Johri, Sunil K Khatri, Ahmad T Al-Taani, Munish Sabharwal, Shakhzod Suvanov, and Avneesh Kumar. Natural language processing: History, evolution, application, and future work. In *Proceedings of 3rd International Conference on Computing Informatics and Networks*, pages 365–375. Springer, 2021.

[18] Łukasz Kufel. Tools for distributed systems monitoring. *Foundations of Computing and Decision Sciences*, 41(4):237–260, 2016.

[19] Felix Lau, Stuart H. Rubin, Michael H. Smith, and Ljiljana Trajkovic. Distributed denial of service attacks. In *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics: "Cybernetics Evolving to Systems, Humans, Organizations, and their Complex Interactions", Sheraton Music City Hotel, Nashville, Tennessee, USA, 8-11 October 2000*, pages 2275–2280. IEEE, 2000.

[20] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[21] Giorgio Alessandro Motta, Bedir Tekinerdogan, and Ioannis N Athanasiadis. Blockchain applications in the agri-food domain: the first wave. *Frontiers in Blockchain*, 3:6, 2020.

[22] Cathrine Paulsen. Revisiting smart contract vulnerabilities in hyperledger fabric. *https://cse3000-research-project.github.io/static/d23cb4583e6d97a1e509eafda859c424/poster.pdf*, 2021.

[23] Pierluigi Di Pilla, Remo Pareschi, Francesco Salzano, and Federico Zappone. Hyperledger fabric attacks mitigation (extended abstact). In *FOCODILE 2022 - 3rd International Workshop on Foundations of Consensus and Distributed Ledgers*, 2022.

[24] Amit Chaudhary published in amitnes. Zero shot learning for text classification.

[25] Atul Singh, Miguel Castro, Peter Druschel, and Antony I. T. Rowstron. Defending against eclipse attacks on overlay networks. In Yolande Berbers and Miguel Castro, editors,

Proceedings of the 11st ACM SIGOPS European Workshop, Leuven, Belgium, September 19-22, 2004, page 21. ACM, 2004.

[26] Atro Voutilainen. Part-of-speech tagging. *The Oxford handbook of computational linguistics*, pages 219–232, 2003.

[27] Qiang Wang and Min Su. Integrating blockchain technology into the energy sector—from theory of blockchain to research and application of energy blockchain. *Computer Science Review*, 37:100275, 2020.

[28] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[29] Wenpeng Yin, Jamaal Hay, and Dan Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *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, Hong Kong, China, November 3-7, 2019*, pages 3912–3921. Association for Computational Linguistics, 2019.

[30] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE international congress on big data (BigData congress)*, pages 557–564. Ieee, 2017.

A. 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=167.94.138.46:45236",
      "sentiment":"very negative",
      "id_log":"MQhep38BYv2pyDWh9xpE",
      "container":{
         "id":"0c06a22e0a5f43b7d2ef9b6bfbfa227ae828a3fd2be0e1ab44e3f46926106640",
         "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