Received 28 July 2024, accepted 22 September 2024, date of publication 10 October 2024, date of current version 22 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3477979

RESEARCH ARTICLE

DBD-Guardian and Privacy-Aware Near Real-Time Cybersecurity Analytics

JOSÉ FRADE, LEONEL SANTOS, AND ROGÉRIO LUÍS DE C. COSTA
Computer Science and Communication Research Center (CIIC), School of Technology and Management (ESTG), Polytechnic of Leiria, Leiria 2411-901, Portugal

Corresponding author: Rogério Luís de C. Costa (rogerio.l.costa@ipleiria.pt)

This work is partially funded by national funds through FCT – Fundação para a Ciência e a Tecnologia, under the Scientific Employment Stimulus - Institutional Call - CEECINST/00051/2018 and in the context of the project UIDB/04524/2020.

ABSTRACT Big Data Cybersecurity Analytics (BDCA) is a helpful tool for cybersecurity maintenance that may support the identification of potential threats. Data preparation for traditional BDCA environments contains several steps comprising data movement, transformation, aggregation, and processing. All these steps take place before data becomes accessible to users. Executing such a workflow may take a reasonable time, which increases significantly with the growing amount and variety of available data sources for analytic operations. As the elapsed time between the actual occurrence of cybersecurity events and data availability for analytical queries grows, BDCA's usefulness decreases. In this work, we deal with near real-time BDCA. We propose DBD-Guardian, a system that runs distributed queries over cybersecurity data sources (e.g., log files) while stored in their original location. DBD-Guardian supports querying heterogeneous unstructured and semi-structured sources by using specialized parsers. Also, as data sources are in their raw format, DBD-Guardian has a component specially designed to deal with sensitive data, providing access to anonymized data. To evaluate our proposals, we prototyped DBD-Guardian and implemented a representative scenario of a small company with several hosts and log files of different types. We also simulated several malicious operations in this scenario and assessed the DBD-Guardian ability to support intrusion identification and enforce privacy protection. We evaluated analytic operations' response time as well. The results proved our solution efficiently supports analytical operations and threat identification and also demonstrated the solution's adaptability to distributed and heterogeneous environments.

INDEX TERMS Cybersecurity, near real-time analytics, privacy, cybersecurity analytics, big data.

NOMENCLATURE

API	Application Programming Interface.
BDA	Big Data Analytics.
BDCA	Big Data Cybersecurity Analytics.
DMZ	Demilitarized Zone.
DNN	Deep Neural Network.
DoS	Denial of Service.
DT	Decission Trees.
FTP	File Transfer Protocol.
HDFS	Hadoop Distributed File System.
IDS	Intrusion Detection System.
IIoT	Industrial Internet of Things.
IoT	Internet of Things.
JSON	JavaScript Object Notation.
KNN	K-Nearest Neighbors.
ML	Machine Learning.
SHA	Secure Hashing Algorithm.
SIEM	Security Information and Event Management.
SVM	Support Vector Machine.
UFW	Uncomplicated Firewall.
UI	User Interface.
VM	Virtual Machine.
VPP	Virtual Power Plant.

The associate editor coordinating the review of this manuscript and approving it for publication was Vicente Alarcon-Aquino.

I. INTRODUCTION

In an increasingly digitalized world, terms such as denial of service (DoS), malware, ransomware, and phishing attacks have become part of everyday life [1]. To cope with the escalating number of cyber threats, users and organizations have resorted to the widespread use of tools such as firewalls, intrusion detection systems (IDSs), anti-malware software, and security information and event management (SIEM) systems solutions. These tools are essential in mitigating the impact of attacks from the various malicious agents present in cyberspace [1], [2]. However, as threats become more sophisticated, tools that can respond to this problem become outdated, allowing malicious actors to carry out attacks without being detected in time [2], [3]. Big Data Cybersecurity Analytics (BDCA) is one of the proposed solutions to this problem [4], [5], [6]. It takes advantage of the latest technologies and promises to deal with the constantly evolving cybersecurity challenges.

BDCA systems have numerous applications, such as enhancing intrusion detection, detecting spamming, spoofing, and phishing attacks, using big data analytics techniques to detect malware and ransomware, and detecting denial-of-service and distributed denial-of-service attacks.

Indeed, BDCA systems rely on the use of Big Data Analytics (BDA) technologies to process data from a wide range of sources related to network traffic data, e-mail records, host and systems configurations, mobile and social media data, and heterogeneous logs files acquired at a wide range of sources, like network management and cybersecurity tools (e.g., firewalls, SIEM, anti-malware software).

BDA technologies and applications are advantageous in many real-world scenarios and have proven valuable in diverse cases. However, BDCA faces unique challenges due to the diverse nature of data sources and the specific requirements of cybersecurity systems. Indeed, proper management of a skilled and multidisciplinary team is essential in data analytics. Smaller companies may opt for the "analytics as a service" model [7], [8], [9]. Other challenges in BDA systems refer to managing and storing large, diverse, and complex unstructured or semi-structured data, requiring intelligent filters, secure and cost-effective data storage, effective pre-processing, adaptive processing, and information extraction mechanisms [7], [10], [11]. Choosing the architecture and tools is crucial, and one should consider the final monetary cost of implementation [8], [12]. Also, achieving both response time and large-scale data querying is an increasing challenge as the number of data sources and the variety of data increases, and specialized performance tuning techniques are usually required in large-scale analytics processing [13]. Hence, selecting the most appropriate technology can be challenging. For example, Spark is often recommended in the literature [14], [15], [16]. Nonetheless, using Spark may not be the optimal solution to the cybersecurity context, as it does not fulfill some of the specific requirements of BDCA, like near real-time decision support.

Response time is critical for cybersecurity, and a BDCA system should support threat detection as quickly as possible, e.g., in near real-time. Existing network anomaly detection solutions and approaches may fail to identify them in real-time for several reasons, including a lack of mechanisms that check if redundant, invalid, and missing data are unnecessarily processed and the need for solutions that mitigate the computational costs of this type of processing [3]. Indeed, SIEMs are a commonly used solution for identifying security incidents. They collect and centralize log data from across the IT infrastructure, transferring, processing, and storing data from various sources. However, their implementation and maintenance can be complex and costly [17].

Ethical and privacy concerns are a significant challenge in big data analytics. The use of unauthorized data, including private and copyrighted information, must be avoided. Data collection should comply with current laws and regulations, such as the General Data Protection Regulation, and avoid abusive collection [7].

Many data sources ingested by BDCA systems contain sensitive information, and privacy requirements must be enforced as defined by existing laws and regulations [18], [19]. User accounts and the IP addresses available in system logs are examples of data that must protected before they can be used [20]. Other examples of sources with sensitive data include social networks, user activity logs, geographic locations, and user network traffic and analysis.

In this paper, we deal with privacy-maintaining near real-time BDCA. We propose *DBD-Guardian*, a solution for BDCA that relies on modules running over a distributed query engine. Our solution uses a novel approach in which dashboard and iterative user queries are executed directly on distributed data sources. A single query can combine multiple sources from a specific host or even combine data sources of different types distributed across the entire environment. One can, for instance, use a single query to access data on the system log files of all servers on the network and combine them with captured network traffic. Data sources (e.g., log files) are queried while still stored in their original locations without requiring their transfer to a centralized repository, reducing data storage and processing costs and providing near real-time access to relevant data. We implemented parsers capable of dealing with the heterogeneity of data sources that allow querying on virtual fields identified in such data sources. Moreover, DBD-Guardian applies anonymization to fulfill privacy-related requirements. We prototyped and evaluated our proposals' performance and threat detection capabilities in a scenario with several threats to hosts of distinct types.

Hence, the main contributions of this paper include (i) a proposal for a solution that provides near real-time access to distributed data for cybersecurity analytics and threats identification, (ii) the identification of privacy-maintenance mechanisms for the distributed query execution context and (iii) a description of the implementation of the solution's prototype and an experimental evaluation on a realistic scenario with several threats.

In the following section, we review some background and related work. Section III describes DBD-Guardian. We present the experimental evaluation in Section IV.

Section V contains the conclusions and outlines future work directions.

II. BACKGROUND AND RELATED WORK

In this section, we review concepts and architectures for DBCA, the use of distributed query execution engines for analytics, and related works from the literature. We also review and organize the main categories of data sources used for BDCA.

A. BIG DATA CYBERSECURITY ANALYTICS

Although the initial definitions of big data relied on the 3V's [21], more recent ones add more characteristics to big data and identify 7V's [22], i.e., *volume*, *velocity*, *variety*, *variability*, *veracity*, *value*, and *validity*. Indeed, BDA involves extracting insights from large and complex datasets and aims to improve the understanding of a system or dataset and predict anomalies or future behavior [11]. However, when used for cybersecurity, the amount of data supporting analytics operations can impact its effectiveness [23].

A BDA system can use statistical, knowledge-based, or artificial intelligence-based tools depending on the volume of data to be analyzed [24]. The Apache Hadoop framework [25] is a popular tool for Big Data processing that relies on a parallel processing architecture known as MapReduce and Hadoop Distributed File System (HDFS), a distributed filesystem for storage. *HBase* is an open-source distributed and NoSQL database on top of HDFS [26] that may be used in such context [27]. *Chukwa* [28] and *Flume* [29] are used alongside Hadoop to monitor distributed systems and collect logs. *Oozie* [30], *Sqoop* [31], *Avro* [32] and *ZooKeper* [33] may be used on top of Hadoop.

Apache Spark [34] is a big data framework that promises better and faster data processing, with the ability to make changes and adjustments at almost any step of the process [14]. It can be integrated with various data storage systems and has several real-world applications, including near real-time anomaly detection in some scenarios [9], [35], [36]. The architecture of Spark consists of four levels: storage, cluster manager, Spark Core, and upper-level libraries, including Spark's *MLlib* for machine learning, *GraphX* for graph processing, Spark streaming for streaming analysis, and Spark SQL for structured data processing [14]. Many studies point to the better performance of Spark versus Hadoop [15], [37], [38].

A BDCA environment has some particular characteristics. Figure 1 presents the typical architecture of a BDCA system [24], which relies on a wide variety of data sources related to cybersecurity, and uses specialized tools for data collection. Typically, the BDCA environment comprises moving collected data to a specific storage, where it is aggregated and processed, and then the results are made available for visualization.

Several options are available to extract data from the network, [39]. For instance, *Wireshark* [40] may be used

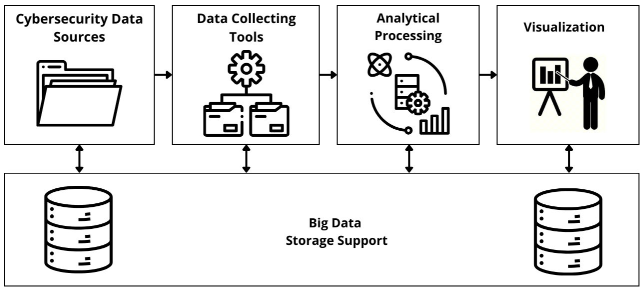


FIGURE 1. Generic architecture of a traditional BDCA environment.

for collection and analysis of network traffic (*Gulp* [41] is well suited for this context as it can read directly from the network), *Nfdump* enables collecting and analyzing networks, and *Nmap* [42] can be used to detect hosts and services in a network.

Tools such as *Flume*, *Sqoop*, *Chukwa*, and *Avro* are useful for data aggregation and transfer in the BDCA environment. They enable the collection and transfer of large amounts of data from various sources and formats [43] and can monitor distributed systems regardless of the programming language used [44].

Big data frameworks such as Hadoop or Spark typically support the data processing phase together with artificial intelligence algorithms. In big data analytics, the most commonly used models are deep neural networks (DNNs), support vector machines (SVMs), and decision trees (DTs) [7]. In the context of BDCA, SVMs, K-Nearest Neighbors (KNN), and DNNs stand out [45]. Indeed, deep learning has been a relevant part of the development of BDA systems and is present in new applications inside many BDCA systems and in the cybersecurity field [46], [47], [48], [49].

The importance of deep learning in BDCA is due to several factors [50]. The first is the capacity of deep learning algorithms to handle labelled and unlabeled data, which is very attractive in big data environments, where data comes in many different forms and with varying structures. Another relevant factor is its ability to discover relationships and hierarchies between unlabeled data that may support data labelling. Finally, deep learning models take great advantage of large volumes of information and are excellent at interpreting unstructured and mixed data. Such capacity they have to deal with the volume and variety of Big Data is another characteristic that demonstrates their usefulness to BDCA [51], [52], [53].

A visualization module is also essential. This component enables understanding the processing results, drawing conclusions, and monitoring the system's operation.

B. DISTRIBUTED QUERY EXECUTION ENGINES FOR BIG DATA ANALYTICS

Recently, some works used distributed query execution engines, such as Presto [54] and Trino [55], for near real-time analytics [56], [57].

Developed by Meta, Presto is an open-source distributed query engine that extracts large volumes of data from various sources and is used in analytic applications and dashboards in large organizations [58]. Presto can efficiently run thousands of concurrent queries on a cluster, simplifies system complexity, and is highly flexible and adaptable. Trino is also a distributed SQL query engine designed to query distributed data over heterogeneous sources [59]. Trino is compatible with HDFS and supports relational and non-relational databases, allowing users to query several data sources using SQL commands.

Trino and Presto offer advantages over Spark and Hadoop for SQL support, interactivity, and wide data source support through their plugin interface. They can also query data in their sources without needing to use data collection, aggregation, and transferring tools, which in the end means that the data consulted and analyzed by these tools will remain in the original data source and will not be copied or transferred to any other device, including the one running the engine. This results in enhanced storage efficiency and management, saving substantial storage space.

EasyBDI [56], [57] runs over Trino to provide high-level abstractions and make data distribution transparent to users. EasyBDI enables fast queries to numerous distributed data sources and provides a high-level abstraction that facilitates query formulation by domain experts.

One main advantage of using distributed query engines is they can query and analyze information directly on the data sources without needing centralized storage, which is usually resource-demanding and time-consuming. Distributed query execution engines execute data integration and aggregation on the fly and may support the execution of near real-time analytics. On the other hand, they access raw data and parsing and transformations are still required. Furthermore, one should prevent accessing sensitive and private data commonly present at BDCA data sources.

C. BDCA DATA SOURCES

Nowadays, the amount of data generated is enormous, but the amount of equipment and systems generating data are equally numerous and diverse. In [45], the authors organize the data sources into three main categories: Network-Based, Host-Based, and Hybrid (a combination of the previous two). Among these categories, Network-Based is the most commonly used (55.74%), followed by Hybrid (13.18%) and Host-Based (6.8%) [24].

SANS [23] presents the results of a survey on institutions and indicates that about 80% of the systems use data sources such as systems, services, and applications (86.30%) and application information (event logs, audit logs) (82.50%). Other of the most used data sources reported in this study were vulnerability management tools (77.60% of users reporting their use) and user behaviour monitoring (with a 41.70% usage rate).

Recent works (e.g., [60], [61]) refer to ‘‘new data sources’’, like reports generated by security tools, web page content, and social networks. Currently, data external to the organizations are also being used in BDCA.

In Figure 2, we organize the BDCA data sources into six categories:

- **Network-based**: These data sources are typically generated by services related to the operation of a computer network or are produced by the flow of communications within that network. Typical examples are network packets, E-mail records, and DNS information. Network-based data sources contain information about the state of all communications, actual configurations of the base services, and the state of the network itself.
- **Host-based**: Various types of information, valuable for security analysis and monitoring, are generated by the activities occurring on the hosts. These kinds of data sources can be produced by the host's operating systems, by user activities while utilizing the hosts, and so forth. System logs, configurations, and executable files are valuable examples of host-based data sources.
- **Inventory and Management**: This class of data source originates from the information available about things related to the management and displacement of IT assets (types, quantities, configurations, etc) and user accounts (roles, permissions, types of accounts, etc) inside an organization.
- **Security Systems**: All the information that comes from the activity of security tools and systems, such as firewalls, SIEMs, and IDSs holds immense value to a BDCA system. These categories of data sources can offer invaluable insights about current threats, alerts, and malicious activities that have been detected, among other types of precious knowledge that can be extracted from this type of data source.
- **User Data**: Some information about users themselves and their activities that can be useful to a BDCA system. All the information encompassed by this type of data source is extracted to delineate what constitutes ‘‘normal’’ user behaviour and then to identify when certain activities (e.g., a login from a location that is relatively far from where the user usually logs in) can be considered suspicious based on that standard. The most common types of user data used in BDCA systems are geographic coordinates, social media, and mobile data.
- **Open Data**: There is valuable information available in open repositories across the internet that can be extracted using open-source tools. By integrating this insight with other categories of data sources, BDCA systems can enhance their analytical capabilities and develop more profound, accurate, and factual conclusions.

D. RELATED WORK

The two main classes for cybersecurity analytics systems are specialized and multipurpose systems. While the former refers to systems whose implementation targets specific tasks, such as detecting phishing messages or malware, the

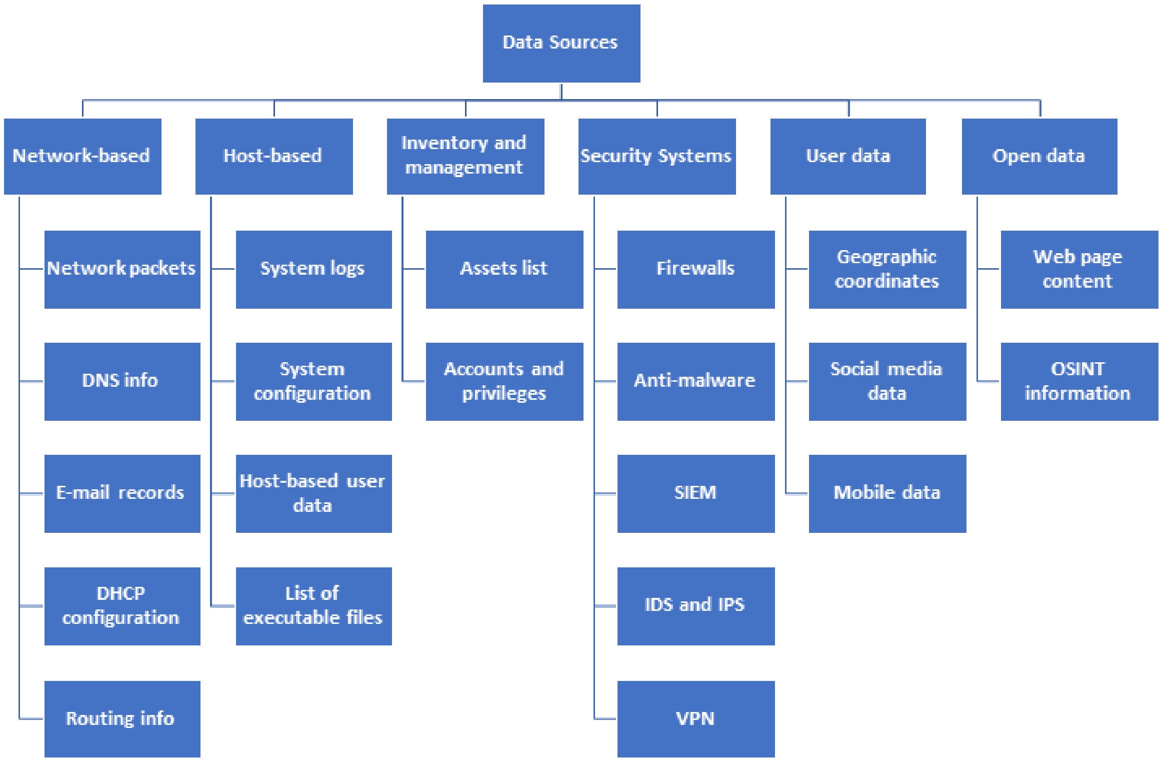


FIGURE 2. Proposed taxonomy of BDCA's data sources.

latter refers to systems that aim to protect all the networks and systems of an organization.

Table 1 summarizes the general characteristics of several works from the literature. Some works are oriented to be applicable to different environments (*single environment*), or developed as a response to a vast array of threats (*multiple threats*). The table also identifies the works that provide the technical details of development and implementation (*provides technical details*) and the ones on which there is a development of a framework (*framework development*).

TABLE 1. Summary of topics and contents covered by BDCA works.

Reference	Framework Development	Provides Technical Details	Multiple Threats	Multiple Environments
[62]	x		x	x
[63]		x	x	
[15]	x	x		
[16]	x	x		x
[64]	x			
[65], [66]			x	x
[67], [68]	x	x		x
[69]	x			x
[70]				
[71]	x	x		
DBD-Guardian	x	x	x	x

In [67] and [68], Ullah et al. address developing a BDCA system that efficiently adjusts itself with minimal to no manual intervention. To this end, the authors propose Quick Adapt and add two additional layers to a BDCA environment. The adapter automatically tunes the configuration parameters of big data analytical frameworks like Hadoop and Spark and executes the BDCA system with the new configurations. Quick Adapt evaluates incoming data against predefined standards and triggers an adaptation process when there is a significant deviation. Some adaptation modes are available, and the mode selection occurs based on the deviation degree. The proposals rely on copying and storing data sources in a centralized repository and processing such data with big data frameworks.

Las-Casas et al. [15] proposed a BDCA system for detecting phishing attacks using Honeypots from all over the world to collect data. Their proposal uses Hadoop and Spark for data processing, HDFS for data storage, and several libraries, including Pig, Hive, GraphX, Streaming, and SparkSQL. The authors evaluated the system's performance by testing its ability to detect spam mail and phishing messages. The authors used the method proposed by [72] for identifying phishing messages among a universe of 19 million with an accuracy of 98.1%. Our work boasts a broader scope than those and does not focus on a single threat type.

In [16], the authors focused on intrusion detection and prevention in corporate networks. Their proposal collects various data, using Cisco's NetFlow tool [73], and data from a local honeypot [74]. The authors implemented their proposals in a company with 200 employees, a thousand daily connections in the honeypot, and 9,600,000 collected flows. The authors evaluated the system's performance with different implementation models and frameworks and concluded that Spark and Shark frameworks would be the most suitable for real-world implementations.

McKeever et al. [64] proposed a model for implementing a BDCA system to protect and integrate European critical infrastructures. The model involves connecting the Security Operation Centers (SOCs) of each critical infrastructure to Security Data Concentrators (SDAs) in a data-sharing network called Critical Infrastructure Security Analytics Network (CI-SAN). The SDAs store data collected by the SOCs and are accessed by Security Analytic (SA) nodes, present all over the CI-SAN, to perform analytical processing. This model allows for a broader and more informed view of the state of critical infrastructure networks, enabling the detection of attacks and anomalies that would otherwise be undetectable.

Empl and Pernul [62] propose a security analytics implementation scheme for small to medium-sized industrial environments using Industrial Internet of Things (IIoT) technologies. The goal is to provide a flexible solution that can adapt to different environments. The proposed architecture enables companies to define and adjust security policies and techniques they want to implement, allowing each company to adapt the operation of the cybersecurity analytics system to its specific needs. A security analytics service module receives large volumes of data from the industrial process, and it is essential to have strong authentication and authorization mechanisms to ensure the protection of sensitive and valuable data. Users can adjust on three levels: analytical techniques, security analytics repository, and security policy repository.

In earlier work, François et al. [63] used MapReduce to detect infected machines throughout the Internet. The authors used MapReduce to implement a cluster allowing the analysis of 77 GB of NetFlows collected daily from 16 million hosts. This early-stage proposal did not deal with near real-time operations, privacy maintenance, and the diversity of threats and data sources.

IBM's QRadar [65] is a SIEM system that collects a large volume of diverse data from various sources and uses it to create a model of normal behaviour for an organization's network and systems to detect possible anomalies. InfoSphere BigInsights is a platform that can be applied alongside QRadar to analyze data collected from various sources, including Qradar's collected data, and alternative sources like social network data. After data collection, InfoSphere BigInsights can apply advanced analytical techniques to the data it collects, enabling it to draw in-depth conclusions about the state of the network [65]. The data is then sent to SIEMs like QRadar to generate alerts that aid security threat identification and mitigation for organizations. Using IBM's InfoSphere BigInsights's big data analytics capabilities with QRadar is a reasonable solution for cybersecurity [66]. Usually, the operation of a SIEM requires vast resources, particularly in processing power and storage, and the QRadar is no exception.

Naseer et al. [69] explores how organizations can use business analytics to exploit analytical information in the cybersecurity incident response. The authors studied how organizations use analytics to respond to cybersecurity incidents, discussing how strong analytics capabilities enhance enterprise security performance and boost strategic and financial benefits.

In [70], the authors present the need for cybersecurity analytics technologies in criminal investigation and discuss how analytic techniques can revolutionize the fight against cybercrime.

Khan et al. [71] presents a framework for improving the cybersecurity of a virtual power plant (VPP) by detecting malicious actors manipulating the VPP's cyber layer operation set-points to violate network stability. The proposed cybersecurity analytic system can warn of an attacker's intrusion attempt by analyzing and predicting anomalies in the voltage reading of the set points in the VPP.

There are also some surveys on BDCA systems in the literature. Ullah and Babar [24] provide a detailed and technical review of the architectural aspects and goals of BDCA systems. Habeeb et al. [3] focus mainly on aspects of the processing component of a BDCA system. Sun et al. [61] focus on the various datasets in this area by enumerating and exploring the nature of these datasets, while in [12], the authors discuss the topic of the scalability of BDCA systems. Mishra et al. [18] provide a short overview of possible applications to big data analytics in cybersecurity and privacy-related issues. In [66], the authors discuss implementations and related challenges of cybersecurity analytics systems in enterprise and academic environments, while in [23], the authors discuss using cybersecurity analytics in organizations.

III. THE DBD-GUARDIAN FRAMEWORK

To provide new real-time analytics capabilities while enforcing the requirements of the regulations for privacy maintenance, we propose DBD-Guardian. It is a framework that provides a dashboard and querying capabilities over distributed data sources for cybersecurity. The framework runs over a distributed query engine, accessing data sources in their original locations and formats. Specialized parsers are responsible for enabling querying on unstructured and semi-structured sources.

A. OVERVIEW OF APPLICATION SCENARIO

DBD-Guardian is adaptable for execution in a variety of scenarios. Figure 3 illustrates the use of DBD-Guardian in a typical small enterprise network. This network has three segments: the Demilitarized Zone (DMZ), the internal network, and the security network. The DMZ is where services meant to be accessible for external clients (e.g., a website) are configured. Such exposed services are isolated from the rest of the network for security purposes. A DMZ features a dedicated firewall with specific rules determining whether an area qualifies as a DMZ. The Internal Network contains all the network services essential for the operation of most small enterprise networks, including email, database management systems (e.g., MariaDB), and an FTP server. Additionally, the hosts are part of this network section. The internal network is separated from the DMZ and the external network by a dedicated firewall, which enforces stricter rules when compared to the ones of the DMZ's firewall. The Security Network is the segment where the security analyst will run the main module of the DBD-Guardian application.

Figure 3 also displays various sample classes of data sources that DBD-Guardian may use for cybersecurity analytics. Local querying components at data sources hosts are responsible for accessing the data sources and transferring queried data to the host where the main DBD-Guardian module runs. A specialized parser is required depending on the data source type. A web-based interface enables the analyst to monitor all other hosts and systems from the network that support query requests from DBD-Guardian.

Table 2 showcases examples of queries DBD-Guardian supports. The table exemplifies the variety of data sources and the ensemble use of sources from multiple hosts or even from different types and single hosts to identify threats.

B. ARCHITECTURE

DBD-Guardian comprises a distributed query engine, integrates a privacy mechanism, and provides a seamless method for incorporating new components. Data is queried at its source, and thus, there is no need to copy data sources into a new host, which reduces the time lapsed between data generation and event detection. Figure 4 presents the main components of the DBD-Guardian distributed architecture.

DBD-Guardian's components may be organized into four groups, namely (i) the hosts and the data sources, (ii) the core modules, (iii) the distributed query engine, and (iv) the external modules.

1) HOSTS AND THE DATA SOURCES

The hosts in Figure 4 represent the machines that will provide data for analytics processing. Each host may store one or more data sources that DBD-Guardian would query. The system must support analyzing large quantities of data and a great variety of data sources stored at the distinct hosts available in the network.

The *Data Sources Manager* is a component responsible for managing configuration information about the hosts and data sources the system accesses. The types of data this component deals with include the name and address of the hosts and, for each host, the name, path, data source type (e.g. TXT, JSON, database log), and parser type (e.g., auth, Syslog, Apache Access) of every available data source.

2) CORE MODULES

DBD-Guardian comprises seven core modules:

- **Privacy Manager**: Privacy Manager ensures that all sensitive data within query results is anonymized. DBD-Guardian relies on a distributed query engine, and the fields to be anonymized are specified in query commands, as in the example of Listing 1. The Privacy Manager has distinct repositories for query commands with and without privacy protection (i.e., the private_queries.json and query.json repositories, respectively).
  Implementing anonymization techniques directly on query commands allows data to be anonymized before it is transmitted from the host to the application. This process occurs when the Distributed Query Engine interprets the query and subsequently fetches the data from the source. At this moment, the anonymization function present in the query is applied to the sensitive information, ensuring its anonymization. With this architecture, sensitive fields that are part of the results of SQL queries are anonymized without the need to anonymize other sensitive data in data sources. Fields to be anonymised are chosen on a per-query basis.
  Considering the queries in the Table 2, examples of data to be anonymized include IP addresses identified in SRC in *ufw.log* or at the beginning of each line in *access.log*, and usernames in messages addresses in *auth.log*. Also, the use of anonymization functions in the SQL commands allows a high level of flexibility in choosing the fields to be anonymized.
  Our current implementation of the anonymization function uses the SHA-256 hash function built-in in Trino, which helps balance the trade-off between privacy maintenance and system usability. Hash is considered an anonymization method of the distortion type [75]. However, one can easily replace the built-in function with a user-created function, which might implement another anonymization type. The system may operate in *Privacy Mode*, applying privacy-protection measures to every query, or in *Non-Privacy Mode*, and show query results without any protection, as access to sensitive data must be provided for authorized users who need to use such data for legitimate purposes, like incident investigation and forensic analysis [17].
- **Authorization Mechanism**: This module collaborates with the Privacy Manager to implement privacy safeguards on sensitive data while ensuring that only authorized users can access this information without any form of anonymization.
- **Query Builder**: This component builds queries that retrieve the desired information in raw or processed formats. It allows for high levels of customization and abstracts technical specificities, ensuring the system's use by business experts. This module accesses a repository of "basic" queries that it rewrites to build the final command the system executes.
- **Data Source Parser**: The Data Source Parser enables unstructured and semi-structured data querying and

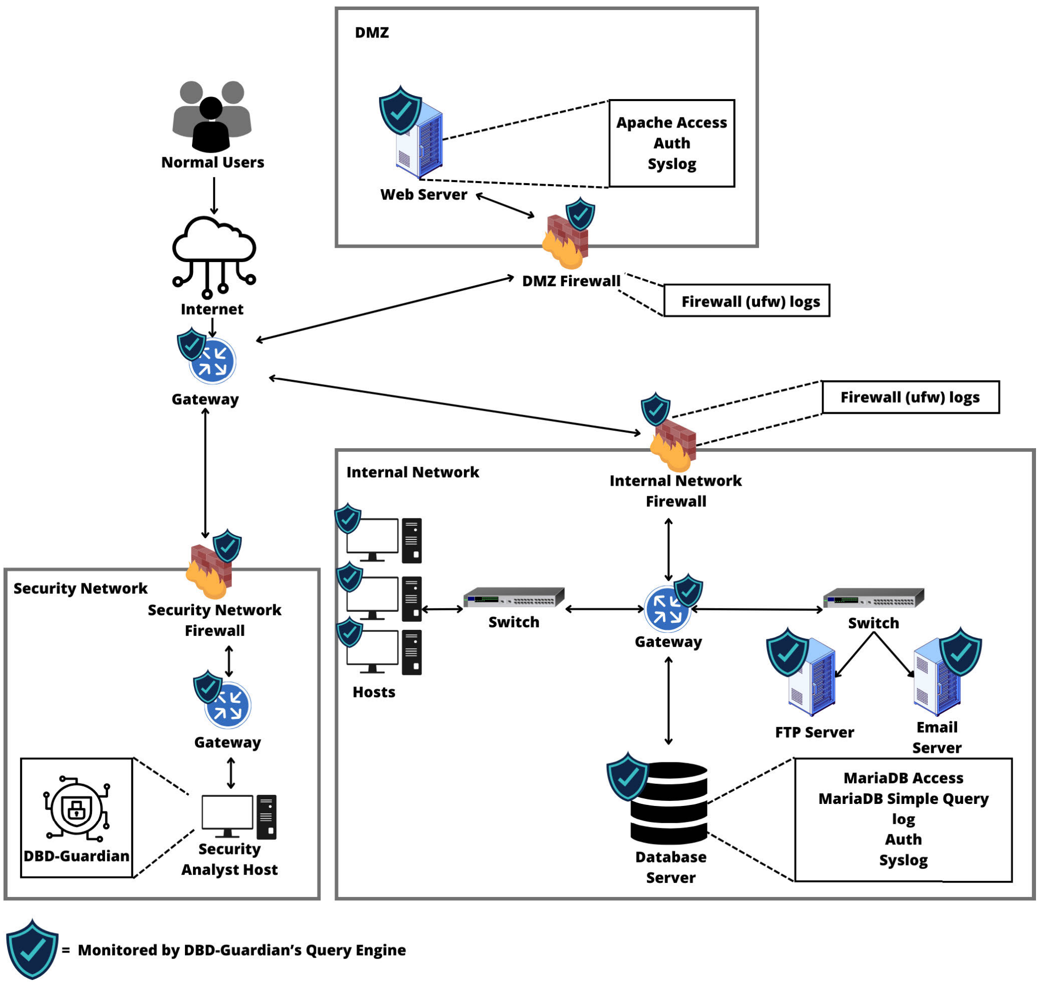


FIGURE 3. DBD-guardian implementation in a small enterprise network.

TABLE 2. DBD-guardian - sample queries.

Query	Data Sources Types	Threats	Hosts/Servers Analyzed
Which IP addresses were most often blocked in the last 24 hours?	ufw.log	DoS, Scanning	One
Which IP addresses had 10 or more failed GET requests in 24 hours, and what is the total number of these failed requests?	access.log (Apache)	Scanning	One
Were there GET requests with SQL commands in the URL?	access.log (Apache)	SQL Injection	One
Which logins are performed on a machine up to 3 seconds after a POST request on the web server that it hosts?	access.log (Apache), auth.log	Forged Auth.	One
Which IPs have the most failed login attempts in the database server?	access.log (MariaDB)	Brute Force	One
What requests were made on the website by the most blocked IP addresses on the firewall?	ufw.log, access.log (Apache)	Sys. Anomaly	One
How many times did the most blocked users logged into the Web server?	ufw.log auth.log	Sys. Anomaly	One
What was the evolution of the total requests made to the database server?	simple_query.log	DoS	One
Which users have the most failed login attempts?	auth.log	Brute Force	One
Are there any users with denied Login attempts in the last day across the network?	auth.log	Brute Force	Multiple
Where in total more than 100 requests in one minute made by the same IP to the hosts?	ufw.log	DoS	Multiple
What was the evolution of the total failed login attempts?	auth.log	Brute Force	Multiple
What was the evolution of the number of incoming packets?	auth.log	DoS, Scanning, System State	Multiple
What was the evolution of the amount of errors registered on the machines?	syslog.log	Sys. Anomaly, System State	Multiple

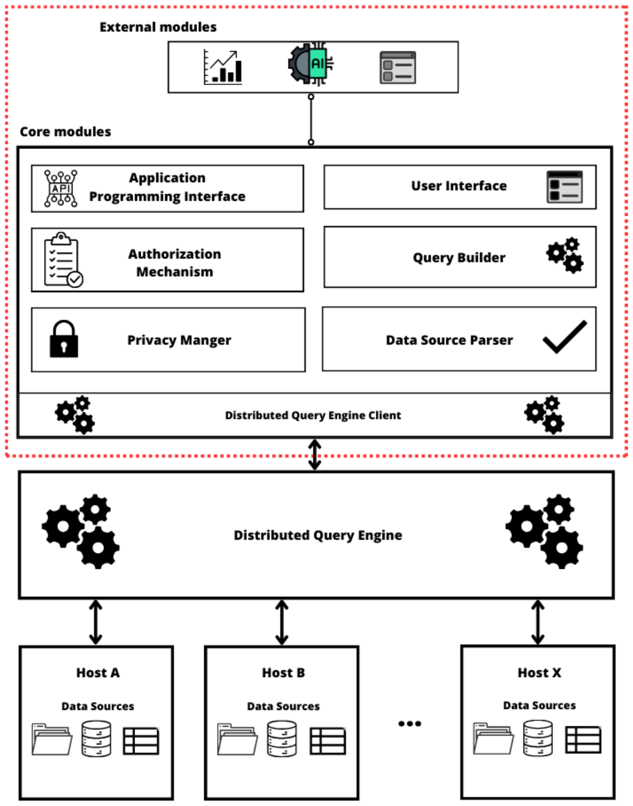


FIGURE 4. DBD-guardian architecture - main components.

analysis. This component is specific for each data source type, making the target source information interpretable by other components. During query building, the system merges parsing instructions into query commands on the fly.

- **Query Engine Client**: This module is responsible for communicating with the distributed query engine, sending queries and receiving their execution results.
- **Application Programming Interface (API)**: An API that adheres to the existing standard (REST) facilitates the system integration with external tools, improving its extensibility.
- **User Interface**: This module provides the interfaces users interact with. It should simplify deriving meaningful insights from the displayed information and provide easy customization.

3) DISTRIBUTED QUERY ENGINE

This component supports both data extraction and analytical processing. The distributed query engine must achieve high performance while handling large volumes of data to provide near real-time access to data and conclusions. In our implementation, we use the Trino distributed query engine for big data analytics.

4) EXTERNAL MODULES

External modules may be integrated with DBD-Guardian through its API to expand the framework's functionalities. Possible external modules include reporting services, specialized dashboard tools, and AI algorithms and models that would process data made available by DBD-Guardian.

C. QUERY BUILDING PROCESS

Figure 5 represents the main workflow of the DBD-Guardian. Once the environment initiates, users may choose the data sources (e.g., operating system log files and auditing logs) to inspect and pre-defined (customizable) queries to add to the dashboard. The system periodically updates the dashboard with queries' execution results. The periodic update of a query resultset may be interrupted if required data sources become inaccessible (e.g., the host or network goes offline). The user can add or remove queries from the dashboard. Dashboard configuration (including queries and corresponding data sources) is stored and will be loaded by the system the next time it starts.

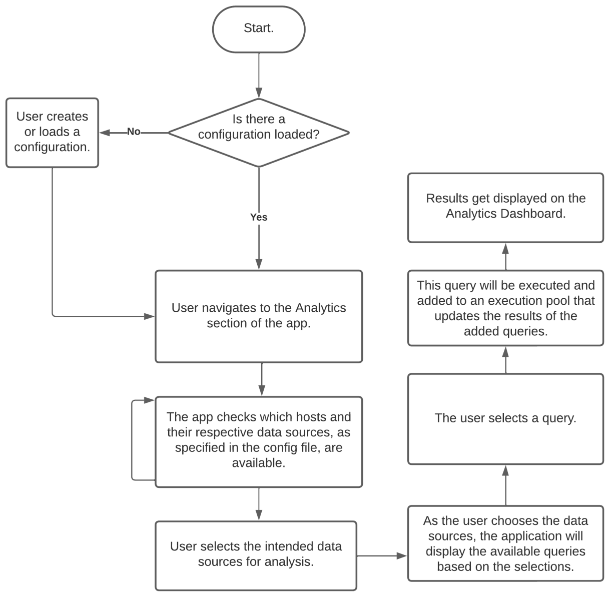

Start.
Is there a configuration loaded?
No
User creates or loads a configuration.
Yes
User navigates to the Analytics section of the app.
The app checks which hosts and their respective data sources, as specified in the config file, are available.
User selects the intended data sources for analysis.
As the user chooses the data sources, the application will display the available queries based on the selections.
The user selects a query.
This query will be executed and added to an execution pool that updates the results of the added queries.
Results get displayed on the Analytics Dashboard.

FIGURE 5. Flowchart of the application.

Analytical querying must provide users with helpful information and insights. DBD-Guardian uses a repository of queries that may be expanded and customized, which enables, for example, formulating queries that help identify specific threats or gather information about the current status of the hosts and services in the monitored environment.

For instance, consider a repository defined using JavaScript Object Notation (JSON). Listing 1 presents an example of a query entry in such a repository. Such entry contains six fields:

- *Title* - the query title/identification.
- *Description* - stores information about a query command, like its objectives and uses.
- *Query* - contains the basic SQL command to be rewritten for execution. This command includes the placeholders "parser" and "file" for parsing and data source path,

respectively. For instance, when querying text files, the "parser" placeholder value is 'storage.txt.file://'. The *anonymization_function* is the function used to anonymize data when running in Privacy Mode.

- *Parse* - An array with the parsing types expected by the query.
- *Type* - denotes the query type in terms of number of sources and hosts (e.g., 1 designates a query over one data source from one host, 2 designates a query with multiple data sources from one host, and 3 designates a query with many data sources from different hosts).
- *Return* - indicates the default presentation type for query results (e.g., lists and chart mode).

```
{
  "Title": "Sample Query",
  "Description": "IP addresses blocked
  by the firewall",
  "Query":
  "SELECT timestamp,
      {anonymization_function}(src_ip)
  FROM ({parser}{file}) as subquery
  WHERE subquery.value
  LIKE '%UFW BLOCK%'",
  "Parse": ["UFW"],
  "Type": "1",
  "Return": "1"
}
```

LISTING 1. Repository of Queries - Sample snippet.

Figure 6 outlines the query building process. When the Query Builder receives a query request, it invokes the Privacy Manager and requests for the active repository of queries, which depends on current privacy configurations. Then, the Query Builder access the adequate repository to get query configuration information (in terms of basic command, parser type, hosts, etc). Parsing information is used to obtain the corresponding parsing instructions from the Data Source Parser. Finally, the Query Builder writes the final SQL command, containing sources' locations and parsing instructions, to be sent to the distributed query engine for execution.

IV. EXPERIMENTAL EVALUATION

To evaluate our proposals, we prototyped DBD-Guardian and performed several tests, including assessing its query execution performance and threat identification capabilities. The source code of DBD-Guardian and data used in the experimental evaluation are available in
https://github.com/
CIIC-C-T-Polytechnic-of-Leiria/DBD-Guardian.

A. IMPLEMENTATION ENVIRONMENT

To implement the core modules, we used the Flask [76] and the Vue.js [77] frameworks. We used JSON files to store application configurations (e.g., host and query configuration and connection information and communication protocol for communicating with the distributed query execution engine). The application menus contain tools to edit such files. In the following, we describe the main implementation details related to each component.

- Distributed Query Execution Engine for Big Data - We used Trino as the distributed query execution engine and the storage connector [78] to enable Trino to query text files. Such a connector, which isn't an official Trino connector, supports querying various file types, including CSV, TSV, JSON, XLS, and TXT. The extraction method varies based on the file type, but the query result is always a table with extracted information organized into columns and rows. For text files, all the lines from the queried file have a single column named "Value".
- Data Sources Manager - The Data Sources Manager repository is stored in a JSON file, whose fields are described in Section III-B1.
- Query Execution Engine Client - We used the Trino Python Client [79] as the query engine client.
- Privacy Manager - The Privacy Manager controls whether the system is in privacy-protection mode. Basic queries are in a JSON file named query.json, while privacy-protected The Query_Priv.json, which mirrors the structure of the first one but contains an SQL command that applies the SHA-256 hash function to the sensible information returned in the output. Figures 7 and 8 exemplify the results obtained using the Privacy Mode. In the former, information is not anonymized, and user login information is available. In the latter, user logins are anonymized through a hash function. Sensitive data are protected, but the distinguishability of individual anonymized entries is maintained.
- Data Source Parser - The Data Source Parser uses a JSON file to store the parsing commands. Every type of data source accessed by the system has an entry on this file. Our current implementation focuses on logs and supports the formats of Syslog, Apache access, Uncomplicated Firewall (UFW), auth, MariaDB access, and MariaDB simple query formats.
- API - The API follows the REST standard and is designed to receive and return JSON Objects.
- User Interface - Through the graphical interface, the user may perform analytical operations, choose and configure data sources and queries, activate the Privacy Mode, create new configurations or load previously generated ones, and review the configured hosts and data sources connection statuses. Users can view all available data sources organized according to the hosts which store them. Figure 9 presents an example of the status information, with two accessible hosts, named *Database* and *Webserver*, on which all the data sources of the former are accessible, but three data sources of the latter are inaccessible. An Analytics Querying

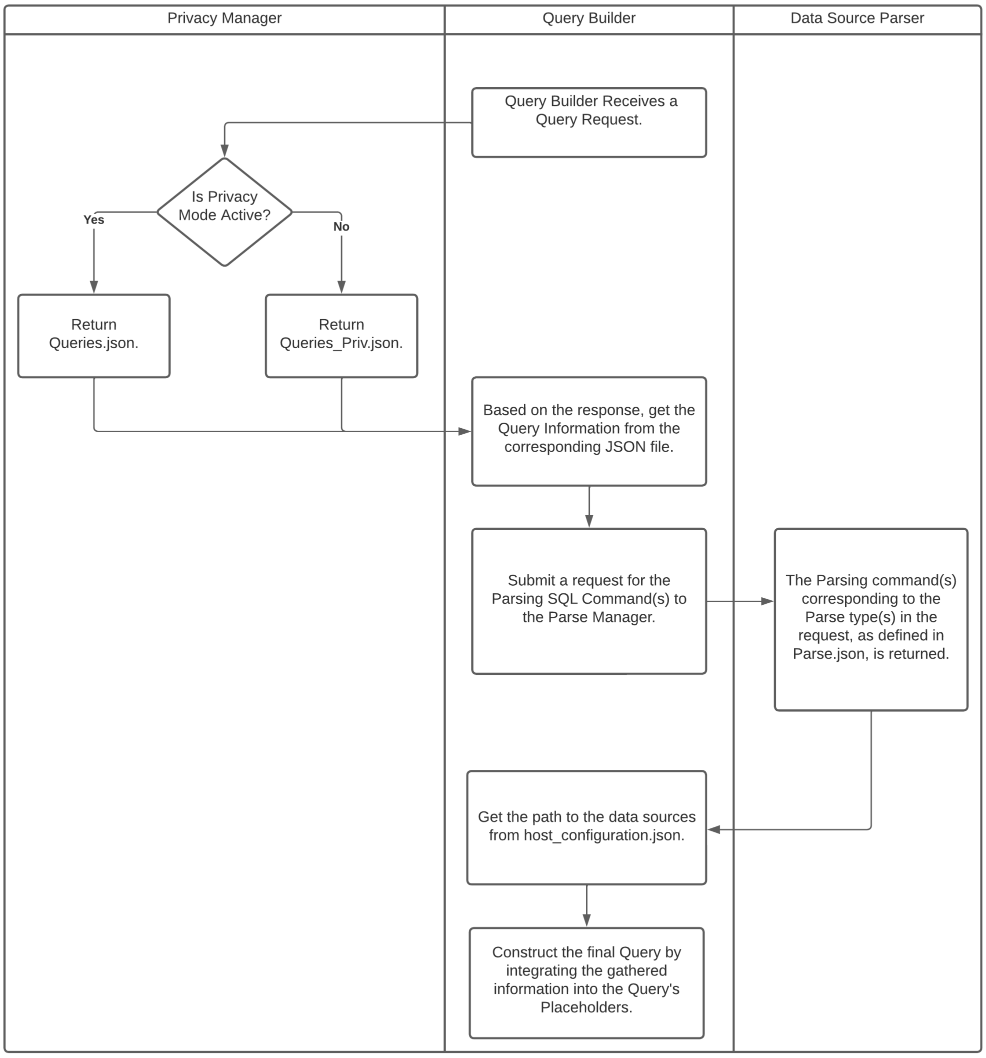
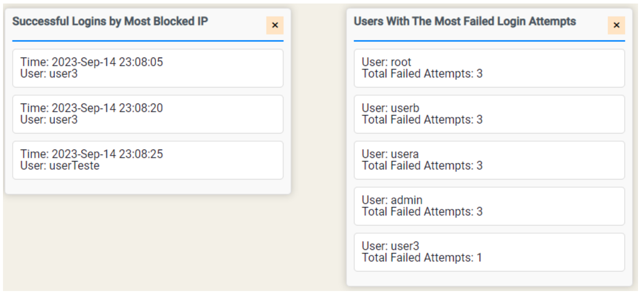

Privacy Manager
Query Builder
Data Source Parser
Query Builder Receives a Query Request.
Is Privacy Mode Active?
Yes
No
Return Queries.json.
Return Queries_Priv.json.
Based on the response, get the Query Information from the corresponding JSON file.
Submit a request for the Parsing SQL Command(s) to the Parse Manager.
The Parsing command(s) corresponding to the Parse type(s) in the request, as defined in Parse.json, is returned.
Get the path to the data sources from host_configuration.json.
Construct the final Query by integrating the gathered information into the Query's Placeholders.

FIGURE 6. Query building operation.



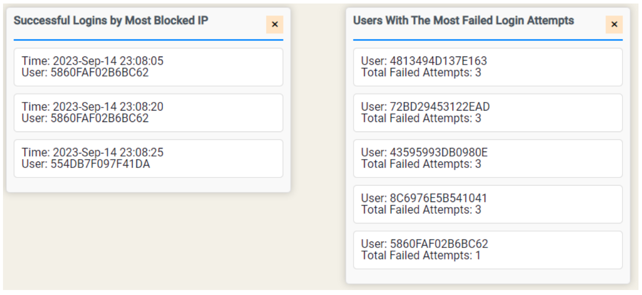
Successful Logins by Most Blocked IP

```
Time: 2023-Sep-14 23:08:05
User: user3

Time: 2023-Sep-14 23:08:20
User: user3

Time: 2023-Sep-14 23:08:25
User: userTeste
```

Users With The Most Failed Login Attempts

```
User: root
Total Failed Attempts: 3

User: userb
Total Failed Attempts: 3

User: usera
Total Failed Attempts: 3

User: admin
Total Failed Attempts: 3

User: user3
Total Failed Attempts: 1
```

FIGURE 7. Query execution results - non-privacy mode.



Successful Logins by Most Blocked IP

```
Time: 2023-Sep-14 23:08:05
User: 5860FAF02B6BC62

Time: 2023-Sep-14 23:08:20
User: 5860FAF02B6BC62

Time: 2023-Sep-14 23:08:25
User: 554DB7F097F41DA
```

Users With The Most Failed Login Attempts

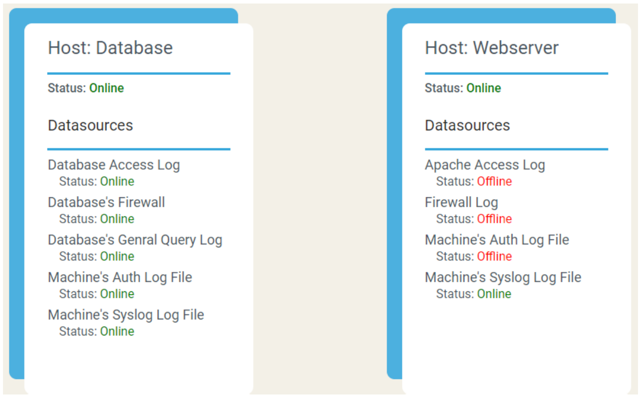
```
User: 4813494D137E163
Total Failed Attempts: 3

User: 72BD29453122EAD
Total Failed Attempts: 3

User: 43595993DB0980E
Total Failed Attempts: 3

User: 8C6976E5B541041
Total Failed Attempts: 3

User: 5860FAF02B6BC62
Total Failed Attempts: 1
```

FIGURE 8. Query execution results - privacy mode.

Wizard automatically displays query options based on the data sources (one or more) the user selects. Available queries depend on several factors, such as the parsing types of the chosen data sources (allowing for various combinations of parsing types), the number of selected data sources, and whether they are present on a single host or distributed across multiple hosts.

B. EVALUATION SCENARIO

Figure 10 presents the evaluation scenario. The network configuration in such a scenario is similar to the one we described in Section III-A and comprises the DMZ, with services accessible for external clients, the internal network, which contains services for enterprise operation,


Host: Database
Status: Online
Datasources
Database Access Log
Status: Online
Database's Firewall
Status: Online
Database's Genral Query Log
Status: Online
Machine's Auth Log File
Status: Online
Machine's Syslog Log File
Status: Online

Host: Webserver
Status: Online
Datasources
Apache Access Log
Status: Offline
Firewall Log
Status: Offline
Machine's Auth Log File
Status: Offline
Machine's Syslog Log File
Status: Online

FIGURE 9. Sample interface - host and data source status.

and the security network which is the segment where the security analyst will run the main module of the DBD-Guardian application. In Figure 10, we highlight the typical communication flows as black arrows and the sequential path taken by the hacker during various attacks across the network as red arrows. The solid red arrows indicate the sequence of attacks leading up to the hacker gaining control of the web server. The dashed red arrows represent the steps the attacker took after gaining control of the web server and until attacking the database server. At the moment of the attack, the DBD-Guardian monitors the DMZ's and internal network's firewalls, the web server, and the database server.

After creating the scenario, we conducted simulations of various cyberattacks by a hacker on the network and generated a substantial collection of log files. These logs contain entries simulating benign network traffic and the traces left by the malicious activities recorded during the simulated attacks.

The attacks simulated in this experiment are:

- **Recognition attack with a Nmap scan**: Nmap [42] is a powerful network scanning tool for network discovery and security auditing. While Nmap is useful for administrators to manage networks, attackers can also use this tool to identify vulnerabilities, open ports, and network services. Hence, Nmap may also contribute to unauthorized access or attacks on the network. The DMZ's firewall UFW log registered data about this event.
- **Recognition attack on the web server**: We simulated Apache's access log entries that could originate from using web server recognition and analysis tools like Nikto, Dirb, and SQLmap. Nikto [80], Dirb [81], and SQLmap [82] are tools for probing the security landscape of websites. Nikto identifies vulnerabilities in web servers, making it a handy tool for attackers aiming to exploit known weaknesses. Dirb assists in discovering existing directories or files on web servers, and attackers could use such a tool to unearth hidden resources. Sqlmap specializes in identifying and exploiting SQL injection points, a tactic attackers use to manipulate a website's database.
- **A successful login via a "reverse shell" exploit**. A reverse shell exploit is a security breach where an attacker establishes a connection from a victim machine to an external system, often under the attacker's control. This connection typically appears as legitimate traffic, enabling the attacker to bypass firewall restrictions. Once the reverse shell is established, it provides a command line interface on the victim machine, granting the attacker unauthorized access. By forging a login through this method, an attacker can stealthily infiltrate the system, execute commands, and exfiltrate data while evading detection by conventional security measures. This simulated forged login left tracks on the auth.log file on the web server machine.
- **A Brute Force Attack to gain access to the database.** A brute force attack is a trial-and-error method to obtain user credentials by systematically attempting all possible passwords until the attacker discovers the correct one. Attackers employ automated software to generate many consecutive tentatives to crack the credentials. This simplistic but often time-consuming attack can be effective against weak password policies, enabling unauthorized access to sensitive systems or data. This attack left traces registered on the database's access log.
- **Suspicious queries on the database** - the attacker simulated several queries, and the DBMS registered such activity on the database's simple_query.log.

C. PERFORMANCE AND THREAT DETECTION EVALUATION

To simulate the proposed scenario, we used two virtual machines running Ubuntu 22.04, each with a one core processor and 6 GB of RAM. The DBD-Guardian main modules run on the host machine. We used Trino version 415. Log files were created using data generated by ChatGPT [83], and their format is close to the typical one used in Ubuntu 22.04.

1) THREAT DETECTION EVALUATION

To evaluate the capabilities of DBD-Guardian to support threat detection and identification, we consider one of the queries at the dashboard is *Number of Requests Made on the Database per Hour*. Figure 11 presents the output of such query execution on the dashboard (which DBD-Guardian periodically updates using data from the database log). It is possible to identify an abnormal amount of traffic on the database on September 17th at 15:00 hours.

Then, to further investigate the abnormal number of requests in the database, one may use the *Failed Login Attempts in the Database* query, which displays events in the last 24 hours. The example in Figure 12 indicates an unusually high number of failed login attempts. Note that sensitive data is protected, and hashes are displayed instead of IP addresses and user logins. However, it is still possible

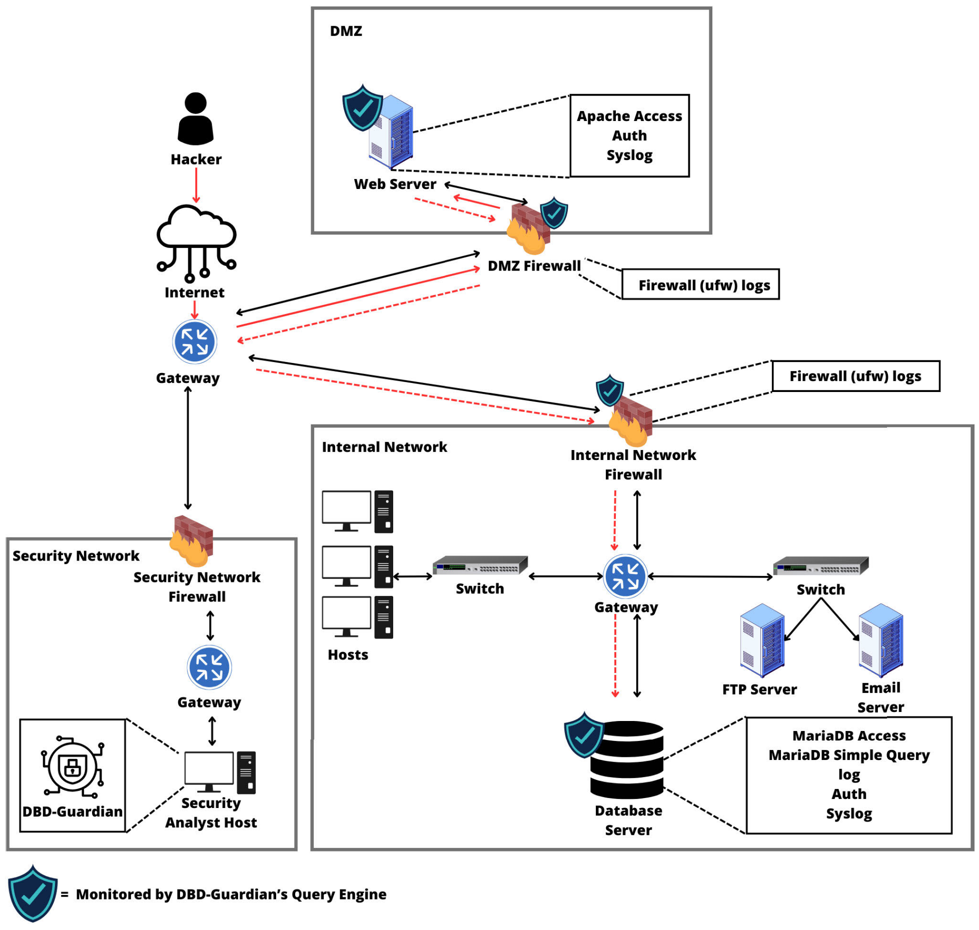


FIGURE 10. Threat identification scenario - network schema.

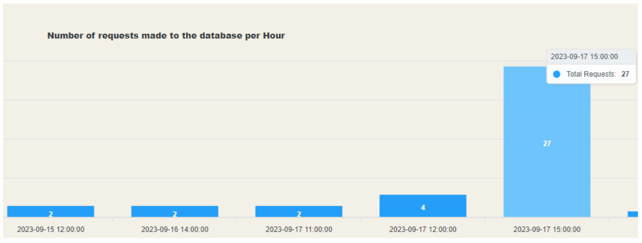


FIGURE 11. Dashboard sample - number of requests made on the database per hour.

to identify that all attempts came from the same address and tried to log in with an identical username.

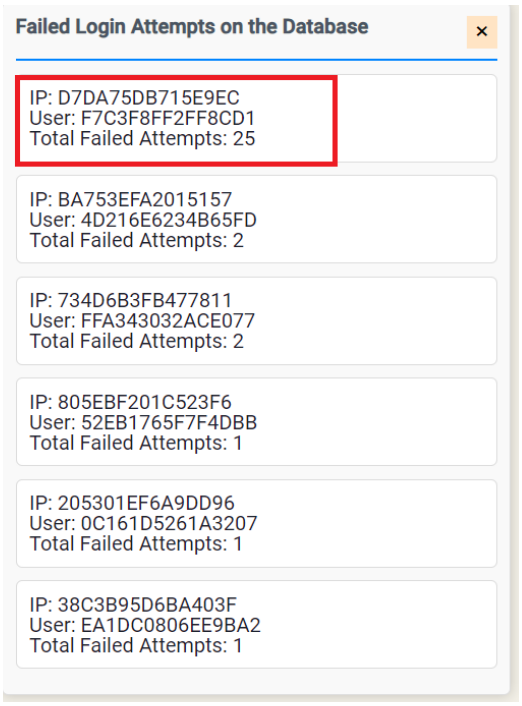
Considering that the database supports a website operation, one may check for any suspicious activities associated with this IP on the machine hosting the Apache web server. The findings are illustrated in Figure 13.

Considering the queries' results, one may conclude that a single IP address was associated with suspicious activities, likely scanning the website for vulnerabilities (such as SQL injection), and an unauthorized login might have occurred. Also, the IP address identified in the web server logs was not the same as the one associated with the questionable activities on the database. To strengthen these hypotheses and gather evidence that abnormal activities might suggest a cyberattack, one may execute additional queries to extract information concerning the IP address most frequently blocked by the firewall. Sample results are presented in Figure 14. One may note that two login operations originated at the most blocked IP address and occurred at the timestamp of the possible unauthorized login. Also, numerous requests to the website were made from that same address.

All these results provide evidence of possible malicious operations. Then, the Privacy mode may be turned off for authorized and legitimate purposes, i.e., incident investigation, which would indicate the address and user logins used by the attacker.

2) QUERY EXECUTION PERFORMANCE EVALUATION

DBD-Guardian accesses data sources on their origin and before any transformation. To evaluate the performance of such an approach, we made a set of preliminary tests that use log files of various sizes and the Trino-storage connector. These files started empty and increased in size in 100 MB



Failed Login Attempts on the Database

IP: D7DA75DB715E9EC
User: F7C3F8FF2FF8CD1
Total Failed Attempts: 25

IP: BA753EFA2015157
User: 4D216E6234B65FD
Total Failed Attempts: 2

IP: 734D6B3FB477811
User: FFA343032ACE077
Total Failed Attempts: 2

IP: 805EBF201C523F6
User: 52EB1765F7F4DBB
Total Failed Attempts: 1

IP: 205301EF6A9DD96
User: 0C161D5261A3207
Total Failed Attempts: 1

IP: 38C3B95D6BA403F
User: EA1DC0806EE9BA2
Total Failed Attempts: 1

FIGURE 12. Dashboard sample - failed login attempts on the database system - privacy mode active.

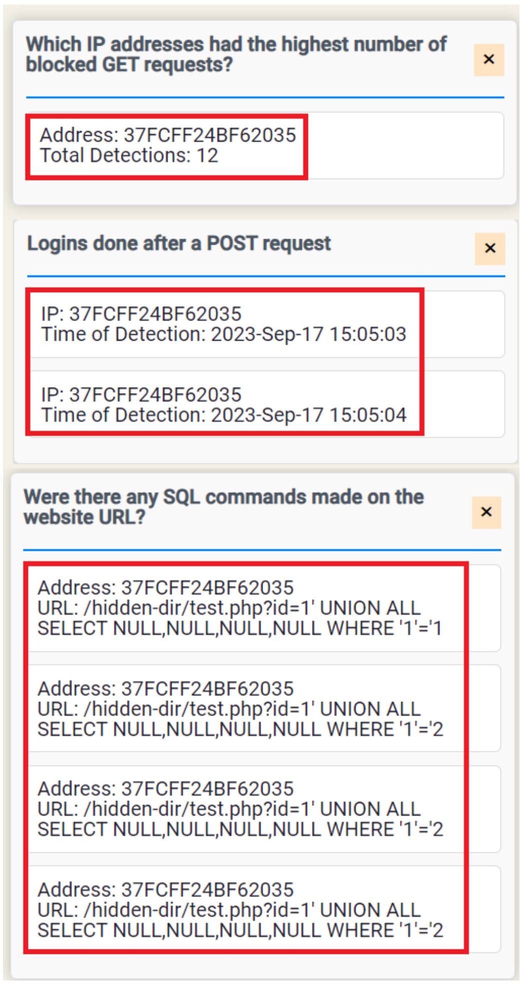
increments, reaching up to 3,000 MB. Each line in these logs adheres to the syslog format.

We measured the query execution time at each log file, repeating the operation three times considering distinct RAM sizes at the machines that store the data sources. Figure 15 presents the average result for each log size. The results are very similar, with a slight difference between the two series indicating that querying times are marginally shorter when using more RAM.

D. DISCUSSION

The evaluation results showed how DBD-Guardian can gather aggregated and detailed information from several sources. While in Privacy Mode, DBD-Guardian anonymized query results and effectively supported the correlation of data resulting from distinct queries while maintaining the privacy of sensitive information. DBD-Guardian could analyze log data within short intervals and support identifying malicious operations in near real-time data.

To implement the extraction of information from new sources or hosts using DBD-Guardian, a developer needs to write a new query and, in some cases, parser instructions without implementing data movement and transformation pipelines. This approach simplifies the development process, reducing complexity and implementation effort and offering significant scalability potential. Also, using a distributed query execution engine leads to reduced storage requirements compared to solutions that copy the data to centralized repositories.



Which IP addresses had the highest number of blocked GET requests?

Address: 37FCFF24BF62035
Total Detections: 12

Logins done after a POST request

IP: 37FCFF24BF62035
Time of Detection: 2023-Sep-17 15:05:03

IP: 37FCFF24BF62035
Time of Detection: 2023-Sep-17 15:05:04

Were there any SQL commands made on the website URL?

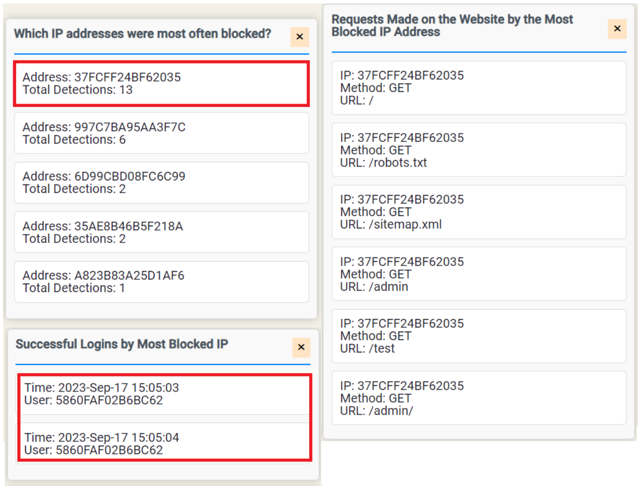
Address: 37FCFF24BF62035
URL: /hidden-dir/test.php?id=1' UNION ALL SELECT NULL,NULL,NULL,NULL WHERE '1'='1

Address: 37FCFF24BF62035
URL: /hidden-dir/test.php?id=1' UNION ALL SELECT NULL,NULL,NULL,NULL WHERE '1'='2

Address: 37FCFF24BF62035
URL: /hidden-dir/test.php?id=1' UNION ALL SELECT NULL,NULL,NULL,NULL WHERE '1'='2

Address: 37FCFF24BF62035
URL: /hidden-dir/test.php?id=1' UNION ALL SELECT NULL,NULL,NULL,NULL WHERE '1'='2

FIGURE 13. Dashboard sample - drill down queries over web server operations - privacy mode active.



Which IP addresses were most often blocked?

Address: 37FCFF24BF62035
Total Detections: 13

Address: 997C7BA95AA3F7C
Total Detections: 6

Address: 6D99CBD08FC6C99
Total Detections: 2

Address: 35AE8B46B5F218A
Total Detections: 2

Address: A823B83A25D1AF6
Total Detections: 1

Successful Logins by Most Blocked IP

Time: 2023-Sep-17 15:05:03
User: 5860FAF02B6BC62

Time: 2023-Sep-17 15:05:04
User: 5860FAF02B6BC62

Requests Made on the Website by the Most Blocked IP Address

IP: 37FCFF24BF62035
Method: GET
URL: /

IP: 37FCFF24BF62035
Method: GET
URL: /robots.txt

IP: 37FCFF24BF62035
Method: GET
URL: /sitemap.xml

IP: 37FCFF24BF62035
Method: GET
URL: /admin

IP: 37FCFF24BF62035
Method: GET
URL: /test

IP: 37FCFF24BF62035
Method: GET
URL: /admin/

FIGURE 14. Dashboard sample - information regarding the blocked IP address by the firewall - privacy mode.

DBD-Guardian's performance highly depends on one of the distributed query execution engines and the parsing operation involved in executed queries. Although in some

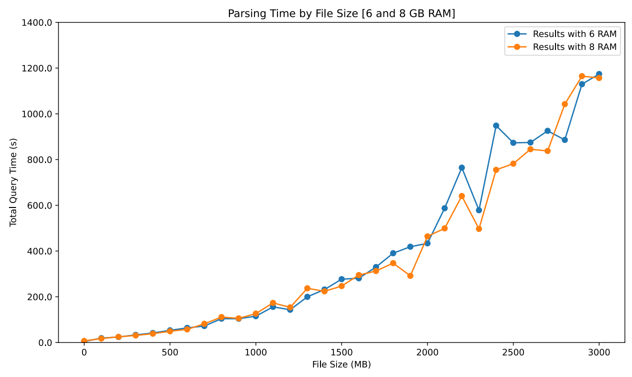


FIGURE 15. Query execution time per file size and RAM size.

production environments, log files might achieve one or more gigabytes, usual cybersecurity maintenance recommendations suggest the implementation of log rotation and a file limit of less than 100 MB [84]. It is important to note that for files of up to some hundreds of megabytes, DBD-Guardian queried the distributed source files in a few seconds, providing near real-time access to data. Still, the solution performance depends on other factors, like the number of hosts and the hardware resources allocated to the distributed query engine. In our experimental environment, the core modules from DBD-Guardian and the main module of the Trino distributed query engine ran on the same virtual machine with just 6GB of RAM, leading to performance degradation when processing distributed files with several gigabytes.

Concerning the network configuration aspect, one might expect that more resources would be required to achieve the same performance if the number of hosts in the network increases. However, the proposed solution uses a distributed query engine instead of copying entire data sources (e.g., log files) to a central host. Therefore, it is a decentralized query solution and querying data from more nodes on the network might benefit from parallelism in distributed query execution. Then, increasing the number of data hosts and sources might lead to lower performance degradation than increasing the size of data sources. Still, we leave a deeper experimental analysis of such aspects for future work.

V. CONCLUSION AND FUTURE WORK

In the last decades, the Internet, mobile devices, and the Internet of Things have become part of everyday life, and cybersecurity has become a major concern. Current BDCA operations involve processing unstructured and semi-structured data from distributed and heterogeneous sources. BDCA systems must support conclusions in near real-time while preserving data privacy requirements.

In this work, we deal with implementing BDCA considering requirements such as privacy maintenance and near real-time responses. We proposed DBD-Guardian, which uses a distributed query engine for big data to perform queries at data sources in their original locations. Our architecture counts on specialized modules for data anonymization. The proposed approach guarantees the maintenance of the privacy of sensitive data but still enables events from different sources to be correlated.

DBD-Guardian proved effective in identifying malicious operations, as we showed in the experimental evaluation. Indeed, In real-world scenarios, even the most sophisticated attacks may leave some trace in the form of log entries. While logs can hold invaluable information about security incidents, analyzing them can be challenging due to their dynamic nature, with constant changes and new entries. Also, log files can be extensive, and their syntax is not always intuitive or easily readable by humans. Our prototype efficiently queries various types of log files in different hosts, providing aggregated data and anonymized information on individual events, supporting straightforward and intuitive analyses of the information in the logs.

Performance assessment showed low response time for log files of typical sizes. Still, the performance of the DBD-Guardian is dependent on the complexity of parsing operations and on the performance of the query execution engine.

In future work, we plan to explore integrating machine learning models and DBD-Guardian through its API to evaluate the impact of anonymization and distributed query execution on models' performance. We also plan to perform an experimental analysis of the performance degradation in different network configurations.

REFERENCES

[1] M. Humayun, M. Niazi, N. Jhanjhi, M. Alshayeb, and S. Mahmood, "Cyber security threats and vulnerabilities: A systematic mapping study," *Arabian J. Sci. Eng.*, vol. 45, no. 4, pp. 3171–3189, Apr. 2020.
[2] S. S. Sekharan and K. Kandasamy, "Profiling SIEM tools and correlation engines for security analytics," in *Proc. Int. Conf. Wireless Commun., Signal Process. Netw. (WiSPNET)*, Chennai, India, Mar. 2017, pp. 717–721.
[3] R. A. A. Habeeb, F. Nasaruddin, A. Gani, I. A. Targio Hashem, E. Ahmed, and M. Imran, "Real-time big data processing for anomaly detection: A survey," *Int. J. Inf. Manage.*, vol. 45, pp. 289–307, Apr. 2019.
[4] R. Alguliyev and Y. Imamverdiyev, "Big data: Big promises for information security," in *Proc. IEEE 8th Int. Conf. Appl. Inf. Commun. Technol. (AICT)*, Oct. 2014, pp. 1–4.
[5] S. S. Nisha, H. Patil, A. Bag, A. Singh, Y. Kumar, and J. S. Kumar, "Critical information framework against cyber-attacks using artificial intelligence and big data analytics," in *Proc. 2nd Int. Conf. Advance Comput. Innov. Technol. Eng. (ICACITE)*, Apr. 2022, pp. 533–537.
[6] T. Y. Win, H. Tianfield, and Q. Mair, "Big data based security analytics for protecting virtualized infrastructures in cloud computing," *IEEE Trans. Big Data*, vol. 4, no. 1, pp. 11–25, Mar. 2018.
[7] I. K. Nti, J. A. Quarcoo, J. Aning, and G. K. Fosu, "A mini-review of machine learning in big data analytics: Applications, challenges, and prospects," *Big Data Mining Anal.*, vol. 5, no. 2, pp. 81–97, Jun. 2022.
[8] D. Goyal, R. Goyal, G. Rekha, S. Malik, and A. K. Tyagi, "Emerging trends and challenges in data science and big data analytics," in *Proc. Int. Conf. Emerg. Trends Inf. Technol. Eng. (ic-ETITE)*, Feb. 2020, pp. 1–8.
[9] A. Ahmed, S. Hameed, M. Rafi, and Q. K. A. Mirza, "An intelligent and time-efficient DDoS identification framework for real-time enterprise networks: SAD-F: Spark based anomaly detection framework," *IEEE Access*, vol. 8, pp. 219483–219502, 2020.
[10] M. Shouaib, K. Metwally, and K. Badran, "Survey on IoT-based big data analytics," in *Proc. 13th Int. Conf. Electr. Eng. (ICEENG)*, Mar. 2022, pp. 81–85.

[11] T. Garg and S. Khullar, "Big data analytics: Applications, challenges & future directions," in *Proc. 8th Int. Conf. Rel., INFOCOM Technol. Optim. (Trends Future Directions) (ICRITO)*, Jun. 2020, pp. 923–928.
[12] F. Ullah and M. A. Babar, "On the scalability of big data cyber security analytics systems," *J. Netw. Comput. Appl.*, vol. 198, Feb. 2022, Art. no. 103294.
[13] R. L. D. C. Costa, J. Moreira, P. Pintor, V. dos Santos, and S. Lifschitz, "A survey on data-driven performance tuning for big data analytics platforms," *Big Data Res.*, vol. 25, Jul. 2021, Art. no. 100206.
[14] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on Apache Spark," *Int. J. Data Sci. Anal.*, vol. 1, no. 3, pp. 145–164, 2016.
[15] P. H. B. Las-Casas, V. S. Dias, W. Meira Jr., and D. Guedes, *A Big Data Architecture for Security Data and Its Application to Phishing Characterization*., New York, NY, USA: IEEE, 2016.
[16] S. Marchal, X. Jiang, R. State, and T. Engel, "A big data architecture for large scale security monitoring," in *Proc. IEEE Int. Congr. Big Data*, Jun. 2014, pp. 56–63.
[17] A. P. Vazão, L. Santos, R. L. D. C. Costa, and C. Rabadão, "Implementing and evaluating a GDPR-compliant open-source SIEM solution," *J. Inf. Secur. Appl.*, vol. 75, Jun. 2023, Art. no. 103509.
[18] A. D. Mishra and Y. B. Singh, "Big data analytics for security and privacy challenges," in *Proc. Int. Conf. Comput., Commun. Autom. (ICCCA)*, Apr. 2016, pp. 50–53.
[19] A. Varanda, L. Santos, R. Luís de C. Costa, A. Oliveira, and C. Rabadão, "The general data protection regulation and log pseudonymization," in *Proc. Int. Conf. Adv. Inf. Netw. Appl.* Cham, Switzerland: Springer, 2021, pp. 479–490.
[20] A. Varanda, L. Santos, R. L. D. C. Costa, A. Oliveira, and C. Rabadão, "Log pseudonymization: Privacy maintenance in practice," *J. Inf. Secur. Appl.*, vol. 63, Dec. 2021, Art. no. 103021.
[21] S. Sagiroglu and D. Sinanc, "Big data: A review," in *Proc. Int. Conf. Collaboration Technol. Syst. (CTS)*, San Diego, CA, USA, May 2013, pp. 42–47.
[22] D. Gupta and R. Rani, "A study of big data evolution and research challenges," *J. Inf. Sci.*, vol. 45, no. 3, pp. 322–340, Jun. 2019.
[23] D. Shackleford, *SANS 2016 Security Analytics Survey*. Swansea, U.K.: SANS Institute, 2016.
[24] F. Ullah and M. Ali Babar, "Architectural tactics for big data cybersecurity analytics systems: A review," *J. Syst. Softw.*, vol. 151, pp. 81–118, May 2019.
[25] A. Hadoop, "Apache software foundation," 2024. Accessed: Oct. 14, 2024. [Online]. Available: https://hadoop.apache.org/
[26] M. Nalin Vora, "Hadoop-HBase for large-scale data," in *Proc. Int. Conf. Comput. Sci. Netw. Technol.*, vol. 1, Dec. 2011, pp. 601–605.
[27] A. Oussous, F. Z. Benjelloun, A. A. Lahcen, and S. Belfkih, "Big data technologies: A survey," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 30, no. 4, pp. 431–448, Oct. 2018.
[28] (Mar. 2021). *Chukwa—Welcome to Apache Chukwa*. Accessed: Oct. 9, 2023. [Online]. Available: https://chukwa.apache.org/
[29] (Apr. 2023). *Welcome to Apache Flume—Apache Flume*. Accessed: Oct. 9, 2023. [Online]. Available: https://flume.apache.org/
[30] (Jan. 2019). *Oozie*. Accessed: Oct. 9, 2023. [Online]. Available: https://oozie.apache.org/
[31] (Jan. 2019). *Sqoop*. Accessed: Oct. 9, 2023. [Online]. Available: https://sqoop.apache.org/
[32] (Jan. 2023). *Apache Avro*. Accessed: Oct. 9, 2023. [Online]. Available: https://avro.apache.org/
[33] (Oct. 2023). *Apache ZooKeeper*. Accessed: Oct. 9, 2023. [Online]. Available: https://zookeeper.apache.org/
[34] (Sep. 2023). *Apache Spark—Unified Engine for Large-Scale Data Analytics*. Accessed: Oct. 9, 2023. [Online]. Available: https://spark.apache.org/
[35] G. D. Ranganathan, "Real time anomaly detection techniques using PySpark frame work," *J. Artif. Intell. Capsule Netw.*, vol. 2, no. 1, pp. 20–30, Mar. 2020.
[36] M. Solaimani, M. Iftekhar, L. Khan, B. Thuraisingham, and J. B. Ingram, "Spark-based anomaly detection over multi-source VMware performance data in real-time," in *Proc. IEEE Symp. Comput. Intell. Cyber Secur. (CICS)*, Dec. 2014, pp. 1–8.
[37] Y. Samadi, M. Zbakh, and C. Tadonki, "Comparative study between Hadoop and spark based on hibench benchmarks," in *Proc. 2nd Int. Conf. Cloud Comput. Technol. Appl. (CloudTech)*, Marrakech, Morocco, May 2016, pp. 267–275.
[38] A. V. Hazarika, G. J. S. R. Ram, and E. Jain, *Performance Comparision of Hadoop and Spark Engine*. IEEE, Palladam, India, 2017.
[39] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, pp. 303–336, 2014.
[40] Wireshark. (2023). *Wireshark*. Accessed: Jan. 18, 2023. [Online]. Available: https://www.wireshark.org/
[41] Gulp. (2023). Accessed: Jan. 18, 2023. *Gulp.js*. [Online]. Available: https://gulpjs.com/
[42] Nmap. (2023). *Nmap.org*. Accessed: Jan. 18, 2023.
[43] P. Joshi, "Analyzing big data tools and deployment platforms," *Int. J. Multidisciplinary Approach Stud.*, vol. 2, no. 2, pp. 45–56, 2015.
[44] P. Grover and A. K. Kar, "Big data analytics: A review on theoretical contributions and tools used in literature," *Global J. Flexible Syst. Manage.*, vol. 18, no. 3, pp. 203–229, Sep. 2017.
[45] I. H. Sarker, A. S. M. Kayes, S. Badsha, H. Alqahtani, P. Watters, and A. Ng, "Cybersecurity data science: An overview from machine learning perspective," *J. Big Data*, vol. 7, no. 1, pp. 1–29, Dec. 2020.
[46] M. A. Amanullah, R. A. A. Habeeb, F. H. Nasaruddin, A. Gani, E. Ahmed, A. S. M. Nainar, N. M. Akim, and M. Imran, "Deep learning and big data technologies for IoT security," *Comput. Commun.*, vol. 151, pp. 495–517, Feb. 2020.
[47] R. Banoth and A. K. Godishala, "Big data analytics for cyber security using binary crow search algorithm based deep neural network," in *Proc. IEEE 7th Int. Conf. Converg. Technol. (I2CT)*, Apr. 2022, pp. 1–5.
[48] E.-U.-H. Qazi, M. Imran, N. Haider, M. Shoaib, and I. Razzak, "An intelligent and efficient network intrusion detection system using deep learning," *Comput. Electr. Eng.*, vol. 99, Apr. 2022, Art. no. 107764.
[49] P. Kanagala, "Effective cyber security system to secure optical data based on deep learning approach for healthcare application," *Optik*, vol. 272, Feb. 2023, Art. no. 170315.
[50] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *J. Big Data*, vol. 2, no. 1, pp. 1–21, Dec. 2015.
[51] Y. Bengio, "Deep learning of representations: Looking forward," 2013, *arXiv:1305.0445*.
[52] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[53] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards AI," *Large-Scale Kernel Mach.*, vol. 34, no. 5, pp. 1–41, 2007.
[54] (Aug. 2023). *Presto: Free, Open-Source SQL Query Engine for Any Data*. Accessed: Oct. 9, 2023.
[55] (Oct. 2023). *Distributed SQL Query Engine for Big Data*. Accessed: Oct. 9, 2023.
[56] B. Silva, J. Moreira, and R. L. C. Costa, "EasyBDI: Near real-time data analytics over heterogeneous data sources," in *Proc. EDBT*, 2021, pp. 702–705.
[57] B. Silva, J. Moreira, and R. L. D. C. Costa, "Logical big data integration and near real-time data analytics," *Data Knowl. Eng.*, vol. 146, Jul. 2023, Art. no. 102185.
[58] R. Sethi, M. Traverso, D. Sundstrom, D. Phillips, W. Xie, Y. Sun, N. Yegitbasi, H. Jin, E. Hwang, N. Shingte, and C. Berner, "Presto: SQL on everything," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Macao, China, Apr. 2019, pp. 1802–1813.
[59] Trino. (2023). *Trino Documentation*. Accessed: Jan. 18, 2023. [Online]. Available: https://trino.io/docs/current/index.html
[60] Z. Li and A. Oprea, "Operational security log analytics for enterprise breach detection," in *Proc. IEEE Cybersecurity Develop. (SecDev)*, Boston, MA, USA, Nov. 2016, pp. 15–22.
[61] N. Sun, J. Zhang, P. Rimba, S. Gao, L. Y. Zhang, and Y. Xiang, "Data-driven cybersecurity incident prediction: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1744–1772, 2nd Quart., 2019.
[62] P. Empl and G. Pernul, "A flexible security analytics service for the industrial IoT," in *Proc. ACM Workshop Secure Trustworthy Cyber-Phys. Syst.*, vol. 16, Apr. 2021, pp. 23–32.
[63] J. François, S. Wang, W. Bronzi, R. State, and T. Engel, "BotCloud: Detecting botnets using MapReduce," in *Proc. IEEE Int. Workshop Inf. Forensics Secur.*, Iguacu Falls, Brazil, Nov. 2011, pp. 1–6.
[64] P. McKeever, M. Allhof, A. Corsi, I. Sowa, and A. Monti, "Wide-area cyber-security analytics solution for critical infrastructures," in *Proc. 6th IEEE Int. Energy Conf. (ENERGYCon)*, Gammarth, Tunisia, Sep. 2020, pp. 34–37.
[65] IBM. (2023). *IBM Security QRadar Suite*. Accessed: Sep. 14, 2023. [Online]. Available: https://www.ibm.com/products/qradar-siem
[66] M. A. Rassam, M. A. Maarof, and A. Zainal, "Big data analytics adoption for cyber-security: A review of current solutions, requirements, challenges and trends," *J. Inf. Assurance Secur.*, vol. 11, pp. 124–145, Jan. 2017.

[67] F. Ullah, M. Ali Babar, and A. Aleti, "Design and evaluation of adaptive system for big data cyber security analytics," *Expert Syst. Appl.*, vol. 207, Nov. 2022, Art. no. 117948.

[68] F. Ullah and M. Ali Babar, "QuickAdapt: Scalable adaptation for big data cyber security analytics," in *Proc. 24th Int. Conf. Eng. Complex Comput. Syst. (ICECCS)*, Nov. 2019, pp. 81–86.

[69] H. Naseer, S. B. Maynard, and K. C. Desouza, "Demystifying analytical information processing capability: The case of cybersecurity incident response," *Decis. Support Syst.*, vol. 143, Apr. 2021, Art. no. 113476.

[70] K. Nallaperumal, "CyberSecurity analytics to combat cyber crimes," in *Proc. IEEE Int. Conf. Comput. Intell. Comput. Res. (ICCIC)*, Dec. 2018, pp. 1–4.

[71] A. Khan, M. Hosseinzadehtaher, M. B. Shadmand, and S. K. Mazumder, "Cybersecurity analytics for virtual power plants," in *Proc. IEEE 12th Int. Symp. Power Electron. Distrib. Gener. Syst. (PEDG)*, Jun. 2021, pp. 1–5.

[72] S. Aggarwal, V. Kumar, and S. D. Sudarsan, "Identification and detection of phishing emails using natural language processing techniques," in *Proc. 7th Int. Conf. Secur. Inf. Netw.*, Sep. 2014, pp. 217–222.

[73] Cisco. (2023). *Cisco IOS Netflow*. Accessed: Jan. 18, 2023. [Online]. Available: https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html

[74] Dionaea. (2023). *Dionaea—Catching Bugs*. Accessed: Jan. 18, 2023. [Online]. Available: https://www.honeynet.org/projects/active/dionaea/

[75] S. Murthy, A. A. Bakar, F. A. Rahim, and R. Ramli, "A comparative study of data anonymization techniques," in *Proc. IEEE IEEE 5th Int. Conf. Big Data Secur. Cloud (BigDataSecurity) Int. Conf. High Perform. Smart Comput., (HPSC) IEEE Int. Conf. Intell. Data Secur. (IDS)*, May 2019, pp. 306–309.

[76] (Jun. 2023). *Welcome to Flask—Flask Documentation (2.3.x)*. Accessed: Sep. 15, 2023. [Online]. Available: https://flask.palletsprojects.com/en/3.0.x/

[77] (Sep. 2023). *Vue.js—The Progressive JavaScript Framework | Vue.js*. Accessed: Sep. 15, 2023. [Online]. Available: https://vuejs.org/

[78] snowlift. (Sep. 2023). *Trino-Storage*. Accessed: Sep. 15, 2023. [Online]. Available: https://github.com/snowlift/trino-storage

[79] trinodb. (Sep. 2023). *Trino-Python-Client*. Accessed: Sep. 15, 2023. [Online]. Available: https://github.com/trinodb/trino-python-client

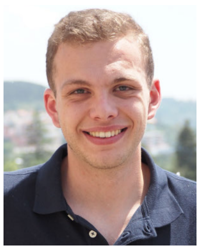
[80] sullo. (Oct. 2023). *Nikto*. Accessed: Oct. 4, 2023. [Online]. Available: https://github.com/sullo/nikto

[81] (Oct. 2023). *Dirb | Kali Linux Tools*. Accessed: Oct. 4, 2023. [Online]. Available: https://www.kali.org/tools/dirb/

[82] (Oct. 2023). *Sqlmap*. Accessed: Oct. 4, 2023. [Online]. Available: https://sqlmap.org/

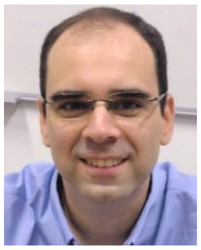
[83] (Sep. 2023). *ChatGPT*. Accessed: Sep. 14, 2023. [Online]. Available: https://openai.com/chatgpt/overview/

[84] Australian Government—Australian Signals Directorate. (2021). *Windows Event Logging and Forwarding*. Accessed: Jul. 7, 2024. [Online]. Available: https://www.cyber.gov.au/resources-business-and-government/maintaining-devices-and-systems/system-hardening-and-administration/system-monitoring/windows-event-logging-and-forwarding



JOSÉ FRADE received the B.S. degree in computer science, in 2022. He is currently pursuing the master's degree in cybersecurity and informatics forensics with the Polytechnic Institute of Leiria. He also made part of the institute's Computer Science and Communication Research Centre, where he has done research on the topic big data analytics for cybersecurity. Currently, he is also a Cybersecurity Analyst, where he performs several tasks, such as vulnerability management and intrusion awareness.

LEONEL SANTOS received the B.Sc. degree in computer engineering, specialization in communication networks and systems from the School of Technology and Management, Polytechnic of Leiria (ESTG), in 2006, and the Ph.D. degree in computer science from the University of Trs-os-Montes e Alto Douro, Portugal, in 2020. He is currently an Assistant Professor with the Department of Computer Science Engineering, ESTG. He is also a Full Member of the Computer Science and Communication Research Centre (CIIC), Polytechnic of Leiria, where he is also a Forensic Computer Expert of the Cybersecurity and Computer Forensics Laboratory—LabCIF and a member of the Scientific-Pedagogical Committee of B.Sc. degree. He has more than 14 years of teaching and research experience in computer engineering, namely in the areas of cybersecurity and computer networks. He has published papers in journals and international conferences in the areas of cybersecurity, information and networks security, and computer science. He has participated in national research and development projects. His major research interests include cybersecurity, information security, networks security, the Internet of Things, intrusion detection systems, and computer forensics.



ROGÉRIO LUÍS DE C. COSTA received the M.Sc. degree in informatics from the Pontifícia Universidade Catlica do Rio de Janeiro (PUC-Rio), Brazil, in 2002, and the Ph.D. degree in computer engineering from the University of Coimbra (UC), Portugal, in 2011. He has over 15 years of teaching experience. He participated in national and international research projects and published papers at leading journals and conferences. He also held technical and managerial positions in software development companies. He is currently a Researcher with the Polytechnic of Leiria, Portugal. His research interests include big data, machine learning, cybersecurity and privacy, and data integration and quality.