applied sciences

MDPI

Article

A Framework for Attribute-Based Access Control in Processing Big Data with Multiple Sensitivities

Anne M. Tall [1,2,*] and Cliff C. Zou [3]

[1] Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816, USA
[2] The MITRE Corporation, Orlando, FL 32817, USA
[3] Department of Computer Science, University of Central Florida, Orlando, FL 32816, USA
* Correspondence: annetall@mitre.org; Tel.: +407-902-3404

Featured Application: Controlling access to a large set of medical and social media data processed in a cloud environment using open-source Hadoop storage and Spark parallel processing.

Abstract: There is an increasing demand for processing large volumes of unstructured data for a wide variety of applications. However, protection measures for these big data sets are still in their infancy, which could lead to significant security and privacy issues. Attribute-based access control (ABAC) provides a dynamic and flexible solution that is effective for mediating access. We analyzed and implemented a prototype application of ABAC to large dataset processing in Amazon Web Services, using open-source versions of Apache Hadoop, Ranger, and Atlas. The Hadoop ecosystem is one of the most popular frameworks for large dataset processing and storage and is adopted by major cloud service providers. We conducted a rigorous analysis of cybersecurity in implementing ABAC policies in Hadoop, including developing a synthetic dataset of information at multiple sensitivity levels that realistically represents healthcare and connected social media data. We then developed Apache Spark programs that extract, connect, and transform data in a manner representative of a realistic use case. Our result is a framework for securing big data. Applying this framework ensures that serious cybersecurity concerns are addressed. We provide details of our analysis and experimentation code in a GitHub repository for further research by the community.

Keywords: attribute-based access control; big data security; database security; parallel processing security

check for updates

Citation: Tall, A.M.; Zou, C.C. A Framework for Attribute-Based Access Control in Processing Big Data with Multiple Sensitivities. *Appl. Sci.* **2023**, *13*, 1183. https://doi.org/10.3390/app13021183

Academic Editors: Marek Pawlicki and Rafał Kozik

Received: 19 December 2022
Revised: 12 January 2023
Accepted: 13 January 2023
Published: 16 January 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Distributed computing and storage are being applied to gain new insights from large datasets that contain information from multiple domains. This type of combined big dataset processing (BDP) supports a variety of applications, such as new medical discoveries, analyses of economic trends, and marketing opportunities. Clusters of interconnected commodity cloud-based systems enable this type of modern data processing. Apache™ Hadoop® is a leading open-source framework for this capability. Hadoop-based services are hosted by cloud service providers such as Amazon, Google, and Microsoft.

In order to realize the analytical advantages, the data security requirements must be addressed while dealing with the velocity, volume, variety, veracity, and value (5-Vs) of big data. A major characteristic of big data is the security risk of a loss of privacy due to reidentification based on compilations of data from many sources. Large datasets are vulnerable to attacks because they may contain highly sensitive data that are otherwise inaccessible from individual data sources.

A framework that provides an approach to protect large datasets is needed. In this paper, we describe our proposed framework approach for BDP security analysis and development. This framework consists of a specific attribute, policy configurations to achieve attribute-based access control (ABAC), and general, system-wide architecture

requirements. Our proposed framework for achieving BDP security consists of three key component actions:

1. Analysis of standards as the fundamental basis for security architecture design. For example, specifically for ABAC, we analyzed Extensible Access Control Markup Language (XACML).
2. Experimentation with a realistic prototype, which we configured using Apache Hadoop (https://hadoop.apache.org/ accessed on 14 January 2023) on Amazon Web Services (AWS).
3. Design of capabilities to ensure operations at an objective secure state, which we identified as including the mandatory use of a secure data analysts' notebook, a data security service layer, and a central management console.
4. We believe our work presents the first framework for a more comprehensive analysis of security for BDP, focusing on the ABAC issues for large datasets with multiple sensitives. Other research on security in BDP has covered a wide variety of specific issues, including homomorphic encryption [1] and developing secure applications that can quickly become operational (DevSecOps) [2]. These advances depend on reliable access control (AC). A variety of alternative approaches to AC are being researched [3], and ABAC is the leading contender. It differs from other methods of access control, such as list-based, in which permission decisions depend upon assigning users to groups or operating system file permissions which designate a file owner, group, and others. Attributes are assigned to users (subject), objects (data files), and data analysis programs. These three types of entity help enable high-fidelity AC.

Our analysis and experiments provide the following contributions:

1. We present specific examples of cybersecurity attacks focusing on access control, which underlies most security services in BDP environments such as Hadoop.
2. We detail the installation and configuration of Hadoop on AWS Elastic Cloud Compute (EC2) instances for security-related analysis and experiments.
3. We provide the results of using a generated representative multiple-sensitivity healthcare and social media dataset in an Apache Spark parallel-processing privacy-preserving data lifecycle.
4. We detail specific BDP configuration recommendations to achieve attribute-based access control (ABAC) in Hadoop using Apache Ranger (https://ranger.apache.org/ accessed on 14 January 2023) and Atlas (https://atlas.apache.org/ accessed on 14 January 2023).

To protect data in BDP environments, system security managers and administrators need to consider and test each component of the ecosystem and the collective environment. Many security features have been added to BDP systems, such as identification using Kerberos and encryption of the data at rest, but the comprehensive perspective of system security needs to be considered along with these individual capabilities. Our framework provides a more wholistic approach. Attackers take advantage of any open port, protocol, and service in the computation, storage, and communication environment to gain a foothold to introduce malware or exfiltrate data. Confidentiality, integrity, and availability need to be applied in a consistent, well-integrated, and layered manner across the environment.

1.1. Motivation

The security of cloud-based BDP, such as AWS Elastic Map Reduce (EMR) (https://aws.amazon.com/emr/ accessed on 14 January 2023), Google Cloud Platform (https://cloud.google.com/dataproc accessed on 14 January 2023), and Microsoft Azure (https://azure.microsoft.com/en-us/services/hdinsight/ accessed on 14 January 2023), is largely based on "castle wall" strategies. It is assumed that everyone with access to distributed cluster computing has full access to all data. Fine-grained access controls that separate data access based on authorizations, roles, and/or data sensitivity is not configured by default in automated cloud-based clusters. Moreover, there is no mechanism to verify that security features have

been implemented and configured and can provide the necessary separation of duty-related protections. This limits the use of clusters to a small number of highly trusted individuals. Ideally, we want to make the data lake available to a wide variety of users with different requirements for analytical processing.

A more wholistic, framework-based approach to security is needed. This includes analyzing standards as the fundamental basis for the design, developing a realistic prototype for experimentation, and then ensuring key capabilities are integrated into the operational system for layered security.

For example, Kerberos has in the past been viewed as the "solution" to Hadoop security. However, with Kerberos, the service ticket issued after authentication (authentication service) by the ticket-granting server (TGS) is valid for a certain period. Hijacking credentials to generate "golden" tickets in Kerberos that never expire is an attack vector that has been exploited in publicly disclosed attacks, such as the Sony attack [4,5]. The threat of this type of attack needs to be carefully assessed, and it should not be assumed that attackers cannot move beyond the perimeter [6,7]. We, therefore, propose our more comprehensive framework approach.

1.2. Big Data Distributed Compute and Storage

The core components of Apache Hadoop and their interactions are shown in Figure 1. The Name Node tracks the location of files stored across multiple Data Nodes and stores their location in the File System Image (FSImage) file. The Resource Manager tracks the execution of data processing through distributed Node Managers. The Resource Manager is a component of the Hadoop Yet Another Resource Negotiator (YARN), introduced in Hadoop version 2. Each Node Manager launches Application Masters (AppMasters) that allocate containers of the Java Virtual Machine (JVM) for parallel process execution. The Job History server and Application Manager oversee job execution across clusters. This architecture enables distributed processing and storage on commodity machines by using programming models such as MapReduce (which is part of the core Hadoop distribution), Apache Spark, and other independently developed open-source tools.

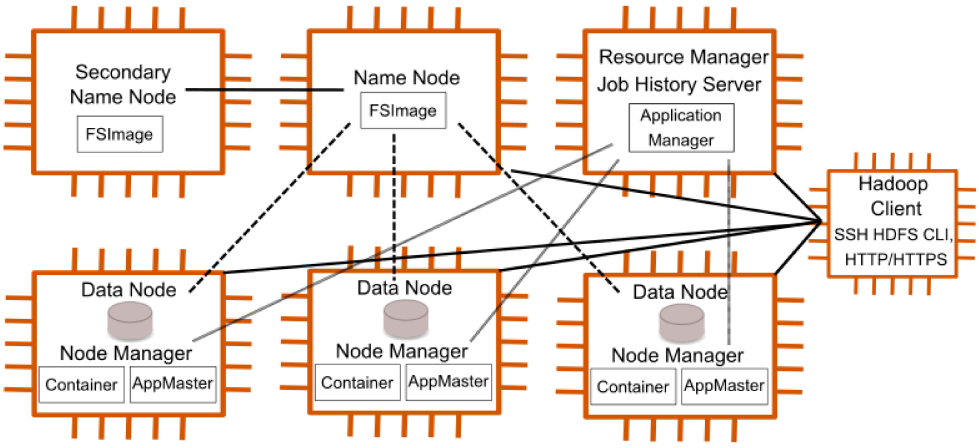

Figure 1. HDFS, YARN, and the Hadoop client component interfaces. Solid lines indicate data and job exchanges during program execution and dashed lines indicate FSImage information exchange.

During a read or write on the Hadoop Distributed File System (HDFS), the Hadoop client interfaces to the core Hadoop components by using the Secure Shell (SSH) and HDFS command-line interface (CLI), a web browser using HTTP/HTTPS, or some other custom TCP interface such as Java Database Connectivity (JDBC). The Name Node that provides the client with the location for reading or writing is assigned the location for writing data blocks that comprise the data file. The client then interfaces directly with each Data Node to read and write the data blocks of the file. Since the client applications used by data-analysts

interface with all Data Nodes, Name Nodes, and Resource Managers, there is potential for a wide attack surface for unauthorized actions.

Applications such as MapReduce and Spark programs are submitted by Hadoop clients to the Resource Manager for execution. Rather than interfacing with each Data Node for program execution, the Resource Manager coordinates execution on behalf of the client. For each client request, it interfaces with each Node Manager co-located with the Data Nodes to initiate an Application Master (AppMaster), which then coordinates the resources of the Data Node (e.g., memory and processors) in JVM containers across the cluster.

The design involves Hadoop clients communicating with every component in the core architecture of Hadoop except for the Secondary Name Node. This node has a limited role in that it periodically executes a checkpoint to synchronize the File System Image (FSImage) and changes captured in edit logs (editLogs) with the Name Node. This enables the capability to restore the cluster if the Name Node FSImage file is corrupted.

From the perspective of cybersecurity, the ability to interface with every Data Node, the Name Node, and the Resource Manager provide Hadoop clients with a large access surface that can cause a significant problem if appropriate access controls are not in place.

New and more reliable Hadoop configurations, such as the Name Node Federation and high-availability configurations, further expand client access to the component interfaces. Multiple copies of files and multiple methods for submitting jobs to the ecosystem further complicate the security of these more advanced Hadoop configurations.

The flexibility of the Hadoop framework, such as exposed application programming interfaces (APIs) that enable multiple types of communication links in the architecture, has enabled the development of a wide variety of independently developed open-source ecosystem software projects. Several of them enable the structured query language (SQL) and SQL-like interfaces, data-ingesting services, job scheduling, provisioning, and management. These capabilities offer easy entry points for data analysts.

A variety of international organizations have contributed to and used the open-source Apache Hadoop ecosystem, including universities, social media companies (Facebook), and cloud service providers (RackSpace, Amazon). Cloudera is a commercial company that is a leading provider of Hadoop support (https://cwiki.apache.org/confluence/display/HADOOP2/PoweredBy accessed on 14 January 2023).

1.3. Attribute-Based Access Control Approach

The overall approach for ABAC is defined in the National Institute of Standards Special Publication (NIST SP) 800-162 as th ABAC trust chain [8]. For the BDP ecosystem, attributes, such as data processing lineage and environmental conditions, are added as part of the AC trust chain, as depicted in Figure 2. When datasets are processed to anonymize sensitive data elements, attributes are updated. Instead of binary, i.e., yes or no, access decisions, a user can be granted access to data after the processing conditions or obligations are met, such as executing an anonymization program and assigning a lower data sensitivity attribute.

The AC trust chain depicts the decision process as a subject, i.e., user or process, requesting access to an object, i.e., the target data or file the subject wants to read, write, or execute. The first step of the trust chain is authenticating the subject based on identity credentials and network access permissions. The AC decision is then initiated based on the attributes of the subject and the authentication rules. We propose that, in addition to the object attributes, the next step of the AC trust chain includes the environmental attributes. In this step, data (object) attributes are used, i.e., metadata or tags, specifically including those that indicate the data lineage, provenance, and privacy-preserving data mining (PPDM) algorithms executed on the data. The rules that define access permissions to data (objects) must include the decision process based upon metadata, i.e., tags. Traditional approaches where users (subjects) are assigned to groups or roles can be incorporated into an ABAC model. The attributes are assigned to the subject roles or groups. The

access permission policies are enforced using the subject, object, and ecosystem, as well as attribute tags, in the final step of the trust chain.

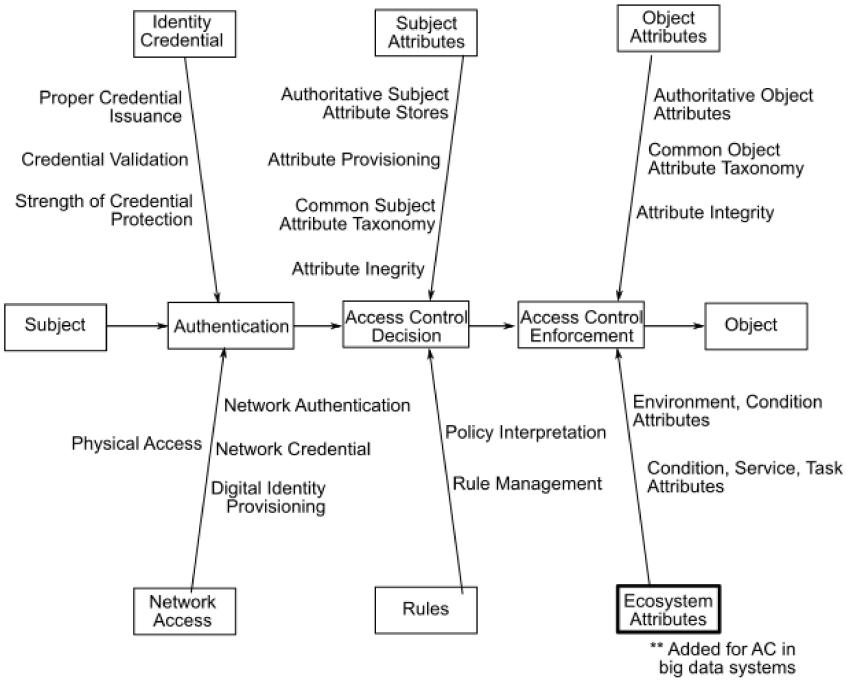

Figure 2. Modified NIST SP 800-162 ABAC trust chain with BDP ecosystem attributes.

As with other security services, AC requires a defense-in-depth approach. Researchers have proposed the application of AC at various points in big data ecosystems based on standard approaches. XACML (http://docs.oasis-open.org/xacml/ accessed on 14 January 2023) is the predominant and de-facto standard for AC. It defines the policy language, request/response scheme, and architecture. Originally published in 2001, XACML is an OASIS standard and is currently at version 3.0. The more recently developed next-generation access control (NGAC) standards address additional areas in securing distributed, multi-owner big datasets. The International Committee for Information Technology Standards maintains NGAC [9].

Although XACML and NGAC policy language standards are widely referenced, they are not implemented in Apache Hadoop ecosystem open-source security projects, e.g., Apache Ranger and Atlas. The reference architectures concepts are basically followed. The XACML and NGAC reference architectures define the necessary functionality and interactions to enforce AC policies. Both architectures specify a four-layer functional decomposition: enforcement, decision, access control data, and administration. The policy enforcement point (PEP) handles user/application requests and interacts with the policy decision point (PDP). The PDP interacts with administrative components, such as the identity provider and policy management components, defined as a policy information point (PIP) and policy access point (PAP). NGAC differs in terms of the interactions of the PIP and PAP with the PDP in the administration of information used to provide additional context for arriving at a decision, such as environmental conditions and obligations.

The standard AC architecture components are shown in Figure 3. Implementation with the retrieval of electronic heath record (EHR) data in the Apache Hadoop environment is highlighted. The sequence of steps that occur when a user requests access to data, e.g., EHR data, is provided as an example. Based upon the user identification (UID), AC policies, and data object attributes, a decision is made to enable a user to view the EHR data. The ecosystem components provide policy enforcement at the boundary using an Apache Knox (https://knox.apache.org/ accessed on 14 January 2023) gateway. Directory services such as lightweight directory access protocol (LDAP) support users/subject AC, and the Hadoop

file system (HDFS) supports AC to data/objects. AC to a process execution can be achieved by controlling access to the Hadoop YARN resource management queue. The administration and management of policy information are achieved using the Apache Ranger and Apache Atlas Hadoop ecosystem components. The sequence of AC enforcement steps and placement of functionality in Hadoop ecosystem components is highlighted in the shaded boxes.

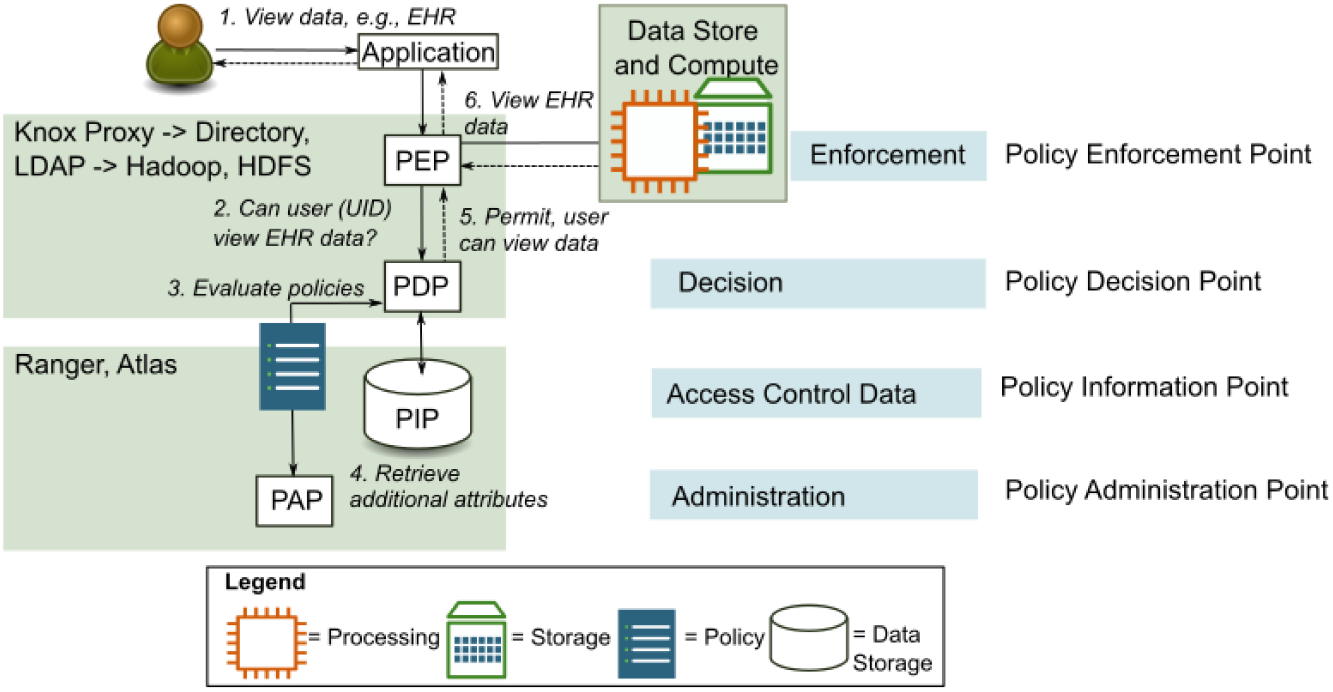

Figure 3. A standard architecture includes multiple points that contribute to the AC process.

There is a lack of consensus on ABAC details, which makes consistent, interoperable implementation a challenge to achieve. For example, a careful design of the rules and definition of the attributes is critical to ensure a clear interpretation of the rules and avoid an explosion in the number of the attributes, i.e., where a large number of attributes are defined and applied to the extent at which they become unmanageable. Alternative locations for implementing AC services in an architecture and avoiding bypasses, and consistent and synchronized implementation are challenging areas explored by researchers. As a result, we believe the first step in designing and implementing a secure BDP is the analysis of the architecture based on standard reference models. An audit of the design should ensure coverage of key functions.

2. Materials and Methods

To verify the recommendations for ABAC in BDP, we designed and conducted a test using a large dataset of messages generated at multiple sensitivity levels. The data are processed in parallel through a series of analytic and privacy-preserving programs to represent a data processing lifecycle. This was hosted on a cloud service provider, i.e., Amazon Web Services (AWS), using storage and computer platforms, Elastic Cloud Compute (EC2), Simple Server Storage (S3), and Infrastructure as a Services (IaaS). This configuration represents a hypothetical yet realistic data analysis environment that may be accessed by users with different permission levels.

Industry and government are investing significant resources into gathering, extracting, transforming, and analyzing large datasets. Without thoroughly tested and trusted BDP security, these types of data lakes are available to only a small group of highly trusted users. The lack of broad use represents a significant potential loss in this investment.

Cloud service providers are helping to enable access to tools to wrangle these large data sets with services such as the AWS Elastic Map Reduce (EMR). Such services provide an operational configuration of BDP frameworks based on open-source software from the Hadoop ecosystem. The current default security configuration is primarily based on security firewalls, proxy servers, and/or virtual private networks at the cluster boundary. These services have only recently integrated fine-grain security services. Service providers are enhancing cloud BDP security based on open-source systems, such as Apache Ranger and Atlas. More complete, robust, and layered security is the operational goal. Currently, the design and configuration of this type of layered security for applications deployed in the cloud currently remain the responsibility of the cloud users.

To further understand the challenges associated with the security of BDP applications running in the cloud, we built an Apache Hadoop cluster with Spark parallel processing, allowing us full access to configuration and log files to achieve a deeper understanding of the security and performance implications. The focus of the experiments the ABAC services for data stored in Hadoop and processed in parallel through Spark applications. This prototype is the second key component of our framework: an experiment that realistically represents a dataset and data processing program with complex security challenges.

2.1. Implementation Strategy

Three technical areas must be specified to implement BDP ABAC: policies, attribute models, and data provenance. Policies are challenging because translating legal, human-language-based, nuanced AC policies into computer programs is complicated. Because ABAC policies are rules based upon relationships between attributes, a consistent attribute definition and management across the big data ecosystem is critical for rule implementation. A variety of attribute models have been proposed by researchers to logically organize and manage attributes in a manner that avoids an explosion of attributes. Data provenance can be considered a specialized, dynamic attribute that tracks data processing over time. In the following subsections, research in these areas is reviewed and summarized.

For each of these three areas, a healthcare use case is considered to further explain and expand upon such challenges. This is provided to illustrate the concept of achieving a fine-grain, dynamic AC. The proposed use case includes a large dataset of mixed EHRs, and social media messages that are accessed by researchers, medical staff, insurance providers, and a large population of patients who are social media users. In this use case, the combined dataset is analyzed for various issues, such as detecting disease spread by using indicators in social media messages. This use case is a representation of a hypothetical "wellness program" that an insurance company might sponsor to improve healthcare outcomes. Figure 4 illustrates the data exchanges between roles in this use case.

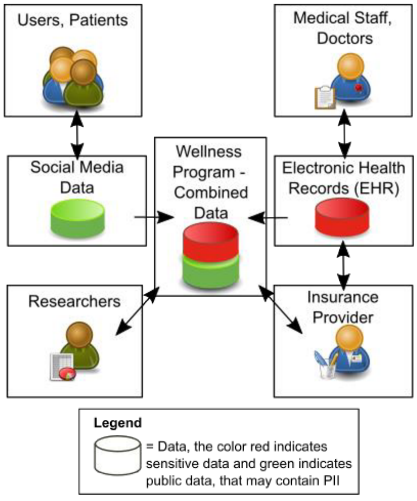

Figure 4. A multi-tenant, multi-level "wellness program" use-case for analyzing BDP security.

2.1.1. AC Policies

The first step in analyzing approaches to AC is considering the representation and implementation of the policies. AC policies are comprised of rules that are typically organized in a hierarchical relationship. The rules define what data users are permitted to access and what actions they are allowed to execute. Privacy requirements based upon Federal laws such as the Health Insurance Portability and Accountability Act (HIPAA) (https://www.hhs.gov/hipaa accessed on 14 January 2023) and industry regulations, such as the California Consumer Privacy Act (CCPA) (https://oag.ca.gov/privacy/ccpa accessed on 14 January 2023) drive the rules at the top of the hierarchy. For example, data that is considered PII or Protected Health Information (PHI) are subject to such legal requirements. Data providers or users, i.e., the data consumer or data owner, also have expectations of data protection and privacy that are translated into rules. The AC rule hierarchy levels are generally as follows:

1. Government regulations;
2. Industry-specified directives (e.g., conflicts of interest);
3. Consumer (data owner) specified directives;
4. Consumer-proxy specified directives.

Translating human-language-based data governance policies into digital language can be a significant implementation challenge for complex, dynamic AC policies. Sen et al. describe the development of LEGALEASE and GROK for building and operating a system to automate governance policy definitions [10]. Human-language privacy policies are implemented in a MapReduce-like big data system. How user data flow among the systems is tracked. Bringing together teams that might not directly interact to define how legal policies are implemented in computer code can be time-consuming. Adding to this cost is the periodic check auditors need to perform to ensure the code continues to comply with the policies. The proposed framework helps automate and speed up the process of defining the data attributes used to control where the data are stored, who can access the data, and for what purpose. Lawyers and privacy personnel encode their policies using the LEGALEASE logic language and then using the GROK mapper, identify the code that might be affected by the privacy policies. Privacy managers can then work with developers, for example, to update only the portion of the code that is affected. This enables more focused updates on the code and data flows to ensure compliance with security policies. The prototype was implemented at a limited scale, and thus there is a need to test the expansion to large complex policies implemented on systems of multiple data-sensitivity classification levels.

Zhioua et al. define a framework for verification of the security guidelines [11]. This covers the software development lifecycle, including the secure coding practices of the Open Web Application Security Process (OWASP) (https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/ accessed on 14 January 2023). In support of AC, the data processing lifecycle is managed using labels and verification checks.

Considering the healthcare use case, requirements for challenging dynamic fine-grain AC policies could present a variety of issues. Fine-grain AC is the ability to control individual subject access to objects, where subjects have different levels of privileges and objects are at multiple classification or sensitivity levels. Fine-grain AC enables more precise control over who has access to which data under dynamic environmental circumstances. In the healthcare use case, policies for emergency situations would allow for AC model flexibility in defining the subject, object, and environment policies and attributes. However, other policies may be more ridged in granting access based on subject role relationships in the absence of an emergency environmental attribute. This could be based upon the legitimate relationships among doctors, medical staff, and patients under the care of (patient of) a provider and/or the hierarchy of medical staff roles where access permissions can be inherited by doctor-designated specialists for certain conditions.

2.1.2. AC Attribute Models

Access security policy rules are expressed and implemented using attributes. There are various approaches to defining attributes, and as a result, defining compatible rules in a federated BDP environment can become complex. The research community has analyzed many approaches to organizing and using attributes in the AC decision process. Their approaches are differentiated based on the use case scenario influencing the authoritative source for the attribute, metadata values, and their dynamics during the data lifecycle. There are overlaps and similarities in attribute schemas that provide the opportunity to identify optimizations. Recently published research on approaches to attributed-based AC policy enforcement is summarized in Table 1. NIST provides guidance and considerations for defining attributes that are reliable, well-informed, and maintainable [12]. The Provenance History Based AC method was the focus of our experiments with the combined healthcare-social media dataset experiment, as described in the following sections.

Table 1. Analysis of ABAC approaches in BDP.

AC Method (References)	Description	Security Observations	Use Case Application
Provenance, History Based [13–15]	The provenance of the data (meta-data) is used as part of the access control decision process.	Supports ensuring execution of privacy preserving programs	Sanitization required to combine and process datasets
Resource Based [16,17]	Permissions are assigned directly to the data. For hierarchically organized data, permissions are inherited by sub-folders from superior folders. Expands upon current file system permissions through centralized management and increased controls on inheritance.	Strengthens previous methods by adding controls on data processing resources	Fine grain dataset control
Task—Role Based [18–20]	Tasks (read, write, execute) are associated with certain roles. Security policies are enforced based on the role a user is assigned and the tasks that role is authorized to perform. Used in healthcare when a patient conducts a medical test and writes data to their medical record.	Strengthens current methods by adding controls to tasks	Fine grain user control
Relationship Based [21–24]	Authorization policies are based upon relationships. Users grant access to information based upon relationships, such as "friends."	Distributed controls to a broad set of users leads to inconsistencies	Online social networks
Role Based, Time Bound [25,26]	Enables temporary, time limited privileges to be provided to users assigned to roles. Enables a just-in-time access model. For example is users access could be limited to business hours.	Risk of unauthorized access constrained by time	Public wifi Access
Object Tagged, Rule Based [27]	Meta-data tags used as part of the access control decision process.	Ensuring integrity of attributes critical to trustworthiness	Dynamic dataset control
Semantic and Ontology Role Based [28]	Addresses mismatches in attribute definitions associated with data from different sources by applying semantics and ontologies as the basis for the access control decision	Technically complex, depends upon accurate inference	Integration of dataset controls across different systems
Content-Sensitivity Based [29,30]	The sensitivity score of data is updated based upon provenance changes. Access to the data is permitted/denied based upon users' access rights to sensitive data scores.	Technically complex, depends upon accurate inference	Needs to infer access rights

Identity standards, such as the Security Assertion Markup Language (SAML) developed by the Security Services Technical Committee of OASIS and implemented in Shibboleth (https://shibboleth.atlassian.net/wiki/spaces/OSAML/overview accessed

14 January 2023), provides standard attribute definitions [31]. SAML is used to express authentication and authorization assertions between a user claim and the response of the contacted system. SAML defines user attributes so that additional information about the users can be provided as part of the sign-on process, thereby supporting service provisioning decisions. The "name" attribute, for example, is expressed in multiple forms in SAML. The XML representation, which has a semantic format, is based upon conventions used in a functional or technical domain or name space, such as X.520, eduPerson (http://docs.oasis-open.org/security/saml-subject-id-attr/v1.0/csprd02/saml-subject-id-attr-v1.0-csprd02.html#eduPerson accessed 14 January 2023), and the National Identity Exchange Federation (NIEF) (https://nief.org/attribute-registry/ accessed 14 January 2023).

Using the healthcare use case as an example, SAML v2.0 for healthcare-defined attributes [32,33] are input to the AC policy decision process. This is illustrated for a big data Hadoop ecosystem in Figure 5.

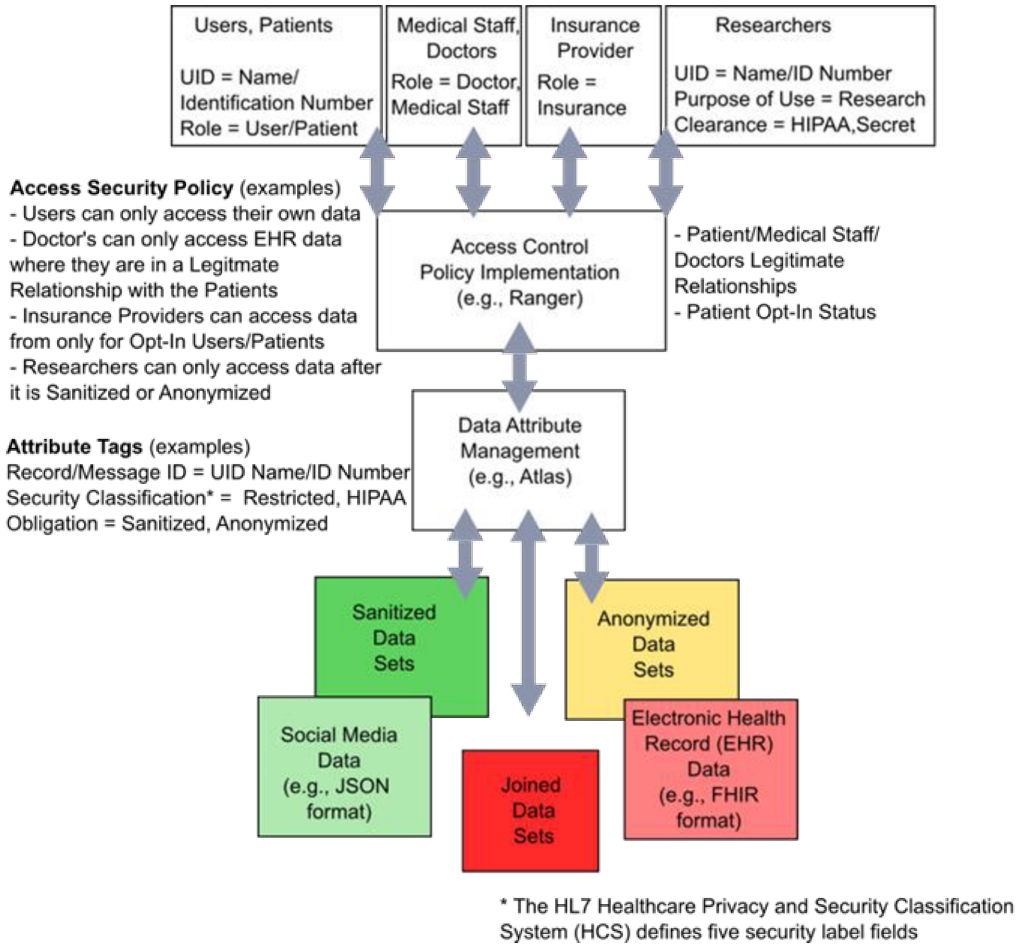

Figure 5. Attributes assigned and managed for AC policy decisions in the healthcare use case. The shades of the color red indicates sensitive data, shades of green indicates public data that may contain PII, and yellow indicates data where sensitive information has been removed.

Encryption provides strong enforcement of AC policies. Attribute-based encryption is a primary research area. Applying encryption in the file system and through proxy gateways are the leading methods. In file system attribute-based encryption (ABE), object data are encrypted with the key associated with a set of attributes. Only subjects, users, or processes requesting access with matching attribute sets have access to the key. Combinations of hierarchical relationships between subjects, objects, and their associated attributes

have been investigated to meet various objectives. The various research projects analyze the efficiency, methods of proof, and alternatives to organize attributes and associated keys. ABE is a security service subsequent to the AC decision process that further protects the data. This area is covered in additional detail in Refs. [34–36].

2.1.3. Data Provenance

Metadata, tags, and attributes that track data provenance are critical to big data systems. Data provenance is defined as the record of the source, processing, and overall lineage of the data. Traditionally, data provenance is associated with audit logs and debugging; however, we propose that it provides an important role in AC. Data provenance can be expressed using a data model, business vocabulary, or other directed acyclic graphic terms. Making big datasets available for analytics requires tracking when processes are executed that reduce the data sensitivity and then updating the data provenance attribute in a trustworthy manner. Research has been published that describes using metadata tags to track processing provenance in this manner (e.g., sanitization history) [37].

Hellerstein et al. [38] introduce three key sources for metadata, which they titled "A-B-C's of data context," i.e.,

A = application or programs run against the data;
B = data source and usage over time (who used it);
C = changes and version history.

Several techniques to protect and reduce data sensitivity have been identified as Privacy-Preserving Data Mining (PPDM) techniques in the ABAC process. Privacy-preserving data protections are integral to AC services and enable linear performance scaling. This implies distributed AC policy decision points (PDP) and the execution of distributed PPDM algorithms [39–41].

Attributes (metadata tags) track application privacy-preserving algorithms on sensitive datasets. The attributes can be used to ensure processing only on authorized computers in the cluster. For example, some policies may require sensitive data storage and processing only in private computing systems, whereas other non-sensitive data in the dataset can be stored and processed in a public cloud service. This type of tag, along with Hadoop rack awareness features and data locality controls, can be used to further control access. This is achieved by incorporating designated data zone considerations, e.g., raw data, private/classified subsets, sanitized subsets, and summary reports, into metadata tags. Saralavedi et al. provide an example of the flexibility that is possible when incorporating metadata tags into big data AC [42].

2.2. Cybersecurity Vulnerability Evaluation

In formulating the prototype experiment as part of our framework, we identified cybersecurity vulnerabilities in BDP technologies specific to Apache Hadoop. This informs residual attack surface evaluation.

Techniques used to attack AC systems are identified using the Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK™) (https://attack.mitre.org/ accessed 14 January 2023) knowledge base, as listed in Table 2. These tactics are based upon actual computer and network attacks documented by the industry and confirmed by MITRE. The Cloud Security Alliance also reports on attack strategies for big data systems and proposes mitigation techniques. Big data set breaches have been reported in several news sources and are anticipated to continue [43–45].

Table 2. Summary of Threats to AC Systems.

Title	Summary	ATT&CK™ Reference ID
Bypass	Accessing data through underlying operating systems or interfacing applications, bypassing the data storage AC. Elevation of privileges by injecting or taking over privileged processes or a trusted connection.	Bypass User Account Control, T1088 Credentials in Registry, T1214 Credentials in Files, T1081 Credential Dumping, T1003 Hooking, T1179 Account Manipulation T1098
Collusion	Leveraging access permissions using a relationship with a third party, exploiting an existing connection or authorized access by an unauthorized user/process.	Trusted Relationship T1199
Inference	Analyzing legitimately obtained data to obtain unauthorized knowledge based upon characteristics of groups or instances	Related to Data Obfuscation, T1001
Unauthorized use of Valid Accounts	Unauthorized use of valid compromised credentials Mitigations include creating an alert and audit log entry of all critical trusted actions and periodically reviewing the logs.	Valid Accounts, T1078 Logon Scripts, T1037 Pass the Hash, T1075 Pass the Ticket, T1097
Timing or Synchronization Attacks	Discovering and using an unauthorized account before the account status is propagated and synchronized across all the systems in the environment.	Related to Account Discovery, T1087
Attribute Proliferation and Assurance	Poor maintenance of or overly complex attributes and metadata tags can lead to errors hard to detect and lead to unauthorized access.	Related to File Deletion, T1107
Denial of Service	Locking out a valid user account or causing excessive process or memory/storage utilization resulting in the disabling of normal operations.	Related to Jamming or Denial of Service MOB-T1067

Approximately twelve Common Vulnerabilities and Exposures (CVEs) (https://hadoop.apache.org/cve_list.html accessed 14 January 2023) are identified on the Apache Hadoop site. The total number on the CVE (https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=hadoop accessed 14 January 2023) site is approximately 45. The number of CVEs is higher because this list also includes items that are closed and not actively being addressed. The list provides a detailed description of attack paths that could be taken to compromise the system, such as the location of sensitive information in temporary files, privilege escalation, security values handling issues, and command injection. Most corrections to address the CVEs require upgrading to a later version of Hadoop.

Security vulnerabilities for all the ecosystem components are also contained in the Hadoop issues tracking system, JIRA. A search for security-related issues in just HDFS and YARN results in over 500 open items. These technical details provide significant insights for specific potential attack paths.

Although there are several security standards to guide the implementation of AC, ABAC, and other related authorization security services, few are implemented in the Apache Hadoop ecosystem components. Kerberos and SASL are used for identification, and SSL can be invoked on HTTP and RPC connections. However, neither XACML nor NGAC are addressed in Apache Ranger (https://issues.apache.org/jira/browse/RANGER-1973 accessed 14 January 2023) or other Hadoop components. This could indicate that there is a lack of maturity in the open-source project to support sophisticated AC methods such as ABAC.

A summary of the analyzed potential attacks is listed in Table 3, and further explained in the subsequent subsections. The vulnerabilities represent paths of entry into the system and include unsecured communication protocols, open connections, and configurations that can be exploited. Cybersecurity attacks are an example of attacker actions and campaigns that have exploited these vulnerabilities. The attack campaigns are known frameworks and libraries of exploitation tools, such as Metasploit, Cobalt Strike, and others detailed on

the ATT&CK website in the techniques and software sections. Researchers have published reports identifying these vulnerabilities in thousands of online Hadoop systems [46–49].

Table 3. Hadoop AC areas, vulnerabilities in the attack path, and patterns of threats posed by attacks.

Access Control (AC) Areas	Vulnerabilities	Threats
Hadoop File System (HDFS)	File system ACs are turned off by default and can be configured in multiple ways	Unauthorized reading and writing to HDFS folders
	SSL/TLS protections on the HDFS WebUI are not enabled by default and can be configured as optional	Man-in-the-middle observation of file system file/folder names, contents, and permissions
Operating System (OS) and Directory Services	OS, Directory, and Hadoop AC configured and managed independently	Misconfigured AC can lead to overly permissive or restrictive access
	Disabled host firewalls and SELinux	Reconnaissance scanning by using NMap, OpenVAS, and other port scanners identifies open ports, protocols, and services leading to unauthorized connections and malware or beacon implants
Resource Management, YARN	Optional configuration of AC lists on job scheduling queues/pools allows all to submit jobs and consume processing resources	Unauthorized job submissions that hijack cluster resources, such as cryptocurrency miner implants and DemonBot
	Proxy servers reuse superuser accounts to prioritize job execution	Submitting malicious jobs, such as ransomware, that encrypt and delete files
Service-level Authorizations	Externally exposed wide port range for RPC and HTTP protocols	Web application attacks, such as cross-site scripting Reverse shell implants through open unauthorized ports, protocols, and services
	Unauthenticated SQL interfaces to the data storage (JDBC)	SQL injection attacks that corrupt or delete data
Management Controls	Different management consoles to multiple ecosystem tools using various ports and login credentials	Attacks are undetected owing to insufficient visibility of the management of system configuration and analysis of logs
	Multiple configuration and log files distributed throughout the cluster that contain sensitive information, such as privileged accounts and passwords.	

2.2.1. Access Control for Hadoop Distributed File System

The HDFS stores data in files and folders by breaking them into blocks and tracking the location of the storage block in the FSImage and editLogs files. The default size of a data block is 128 MB, and this is configurable. These blocks are replicated, typically three times, to avoid data loss and support high availability.

The permission settings for files and folders in HDFS are based on the POSIX model. However, there are certain differences because there is no concept of executable files, i.e., program files are stored outside the HDFS. If the HDFS is configured to conduct permissions, check for a file or directory accessed by a client process, and the traditional, owner, group, and other permissions are checked. If a file permissions check fails, the client operation fails regardless of whether it is a simple command-line interface request or a job execution.

If the property in the core-site.xml file is not set to check authorizations, only the account under which Hadoop is run, i.e., the Hadoop superuser account, can access the files and directories. In this situation, proxy servers are typically used to map all permitted users to the privileged superuser.

The contents of the file system and the status of block replication in the Data Nodes are displayed through a web browser-based graphical user interface (WebUI). This interface, as well as all HDFS files, reads, and writes, can be executed over the Secure Socket Layer (SSL)/Transport Layer Security (TLS) if configured. By default, Hadoop exchanges data in

clear text. This exposes the data exchanges to man-in–the-middle attacks. A wide variety of unauthorized data reads and writes can occur if the security settings are not enabled.

Although the basic capabilities for managing and controlling the HDFS are in place and are based on the familiar POSIX format, they are not enabled by default. The lack of default security configurations can result in many mistakes in AC.

2.2.2. Operating System and Directory Access Control

User authentication and AC in Apache Hadoop can occur at the level of the operating system (OS) (Linux) or over the network by a directory server by using the Lightweight Directory Access Protocol (LDAP) if configured. This is not configured by default and can result in AC errors.

Another area of concern for OS security is that the installation recommendations for Hadoop can disable the Linux firewall (IP tables for IP versions 4 and 6) and kernel security (SELinux). These powerful Linux security tools have a strong evaluative heritage and level of trust. A primary function of these tools is to tightly control the ports, protocols, and services that are externally focused and the programs that are authorized to run at a given level on a host. Correctly configuring these services requires a complete understanding of the executing software.

When the host-based firewalls are disabled, hosts can be subject to reconnaissance scanning by open-source tools such as Network Mapper (NMap) and OpenVAS. Reconnaissance provides attackers with indicators of what is enabled. They can then derive potential weaknesses in the hosted software for further attacks, e.g., Christmas Tree attacks. Flooding listening ports with active protocols can also lead to a denial-of-service scenario.

SELinux also requires a strong understating of the hosted software and the privileges needed; for example, controlling the ability to read and write critical files and activate and deactivate critical OS processes. SELinux can help prevent the introduction of unauthorized programs that exhibit unacceptable behaviors, such as privilege escalation attacks, to activate ports/protocols to communicate with a remote system, e.g., in beaconing attacks. The fact that many mature Linux distributions, such as Red Hat and CentOS, have SELinux enabled is a testament to its value.

2.2.3. Resource Management and YARN Access Controls

The Resource Manager was introduced in Hadoop 2.0 as a component of YARN. It oversees the division of processing load on Data Nodes by using the Node Manager, Application Masters, and JVM container daemons.

YARN obeys the file permissions settings of the HDFS and uses the identity of the user (by default) to interact with the file system. YARN considers only the files that the user owns to execute the YARN program and does not perform any privileged action. It is possible, however, to specify a different user such that the YARN Resource Loader interacts with HDFS using their rights. When Hadoop is accessed by multiple users, separate instances of the Resource Loader should be created (one per user) instead of assigning additional permissions or groups to one user. This ensures that ACs in the HDFS is applied per user. However, if impersonation is used with a proxy server, the Resource Loader might (and will typically) return restricted files that should not be seen by the user.

To control permissions to submit jobs to YARN, ACs can be configured in the context of queues and pools for job submission schedule. For example, when Spark users submit their jobs, they are added to the job scheduler after authentication and AC-related decisions. YARN has three types of schedulers: a capacity scheduler (default), first-in-first-out (FIFO), and a fair scheduler. By default, the permissions to submit jobs in the yarn-site.xml configuration file are open (e.g., set to all using an asterisk).

A more user-friendly and flexible approach to managing job scheduling ACs, rather than configuring XML files, is for the system administrators to use Apache Ranger. There are Apache Ranger YARN plugins that can manage policies on users that are allowed to submit jobs to YARN. These policies enforce rules on who can submit to the YARN queue.

When the AC of the YARN scheduler is not configured, the cluster can be subject to a wide variety of unauthorized job submissions. Several unauthorized cryptocurrency mining programs have been detected in Hadoop clusters, such as DemonBot. A more destructive threat is ransomware which can destroy a large-scale cluster dataset by encrypting the data and withholding the decryption key [50,51].

2.2.4. Service-Level Authorizations

Client services need to incorporate identification, authentication, access control, and data confidentiality (i.e., encryption) into BDP environments. This applies to all actions, including submitting jobs (Apache Spark and MapReduce), reading and writing files, and interacting with the web-based graphical user interfaces (Web GUIs). TCP/IP protocols encase the remote procedure calls (RPCs) that contain the client-to-server (Name Node, Resource Manager, and Data Node) and server-to-server communications. For example, Data Nodes send heartbeats to report their health status to the Name Node and use an additional TCP/IP data transfer protocol to exchange data blocks [52].

Client applications for the Hadoop ecosystem include Apache HUE (https://gethue.com/ accessed 14 January 2023), Zeppelin (https://zeppelin.apache.org/ accessed 14 January 2023), and HDFS CLI, which can incorporate security services. If configured, Kerberos enforces client authentications (identification). Data Nodes use the Simple Authentication and Security Layer (SASL) (https://www.rfc-editor.org/rfc/rfc4422 accessed 14 January 2023) to negotiate authentication (e.g., Kerberos) when using the data transfer protocol to exchange data blocks.

Authorizations are defined based on access control lists (ACLs) contained in the hadoop-policy.xml configuration file or as defined in a component of the ecosystem external to the core Hadoop components, e.g., Apache Ranger, as described below.

If configured, the SSL/TLS secure information exchanges. For Hadoop clients and servers, this protects the RPCs. SSL/TLS also protects the web interface used by servers to expose their status in that it can be configured with HTTPS. This includes, for example, the Web GUI for the Name Node and the Resource Manager.

In addition to ACLs, Hadoop components that provide a JDBC-SQL interface, e.g., Apache Hive, have authorization services that must be configured. For example, executions of the GRANT and REVOKE commands and SQL standard authorizations were added to later versions of Hive.

The ACs for these various protocols that are used by different components are managed through several configuration files. They can be very complex and difficult to translate to modern AC models, such as ABAC. The use of a management console such as Apache Ranger helps centralize this configuration and ensure consistency. This is instrumental for achieving security-related objectives.

Eternally exposed protocols such as HTTP, RPC, and JDBC-SQL can be subject to a wide variety of attacks if not secured, even when running on a high port range. For example, web application attacks include SQL command injections, cross-site scripting (XSS), web shell implants, and injections of brute force commands.

2.2.5. Management Consoles

The complexity of integrating different components of the BDP ecosystem is handled by management consoles such as Apache Ambari (https://ambari.apache.org/ accessed 14 January 2023). However, they largely do not directly address security configurations. Most Hadoop consoles focus on performance-related statistics. Separate and independent management interfaces, such as Apache Ranger, have been developed to fulfill this role. Without a central view of the security status of Hadoop, the system's security managers can be blind to attacks. Unauthorized actions may only be detected after a review of complex log files. Amazon has recently announced that EMR will be integrated with Apache Ranger [53].

A commonly documented weakness (CWE-778; https://cwe.mitre.org/data/definitions/778.html accessed 14 January 2023) is that when security-critical events are not logged properly, detecting and hindering malicious behavior becomes much more difficult and may hinder forensic analysis once an attack succeeds.

Figure 6 shows the use of Apache Ranger and Apache Atlas to enforce and manage security policies through the core Hadoop components. The interconnection of a directory server to manage user authentication and AC is included here.

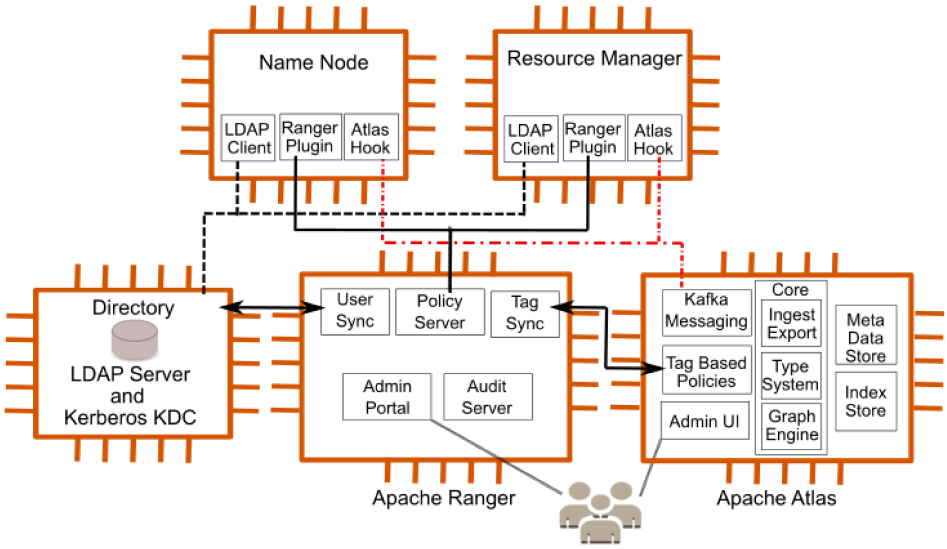

Figure 6. Apache Ranger and Atlas interfaces to HDFS Name Node and YARN Resource Manager. Directory information exchanges indicated by black dashed lines, Ranger policy information exchanges are indicated by solid black lines, Atlas attribute tag information exchanges are indicated by dashed red lines, and synchronizations between the directory, Ranger and Atlas are indicated by double headed arrows.

The actions and information exchanges in the HDFS environment are depicted in the sequence diagram shown in Figure 7. Attributes managed by Atlas are used by Ranger in managing ABAC policies. These policies are enforced in the Name Node and Data Nodes. When jobs are submitted, tasks executed, and tasks and jobs completed, the ABAC policies are enforced by each component. Attribute updates for generated HDFS files are then sent from Atlas to Ranger. Subsequent sections highlight the key capabilities of these projects in supporting security management.

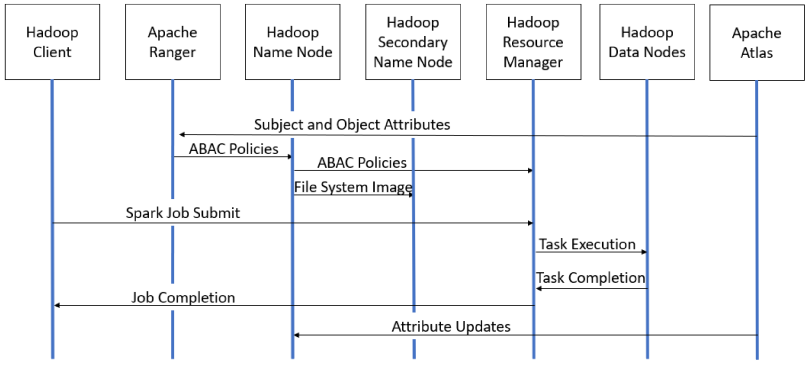

Figure 7. The sequence of attribute information exchange in job execution.

Apache Ranger

The Apache Ranger framework is a collection of software components that can be flexibly configured to monitor and manage data security. It includes a WebUI to administer security settings, including fine-grained authorizations (AC) in such Hadoop components as Hive, HDFS, and YARN. It helps with centralized audit tracking for each user action. The components of Ranger include an administrative portal, policy storage (in MySQL or another database), audit log storage (in Solr, HDFS, or another database), synchronization with user accounts in a directory (User Sync), synchronization with metadata tags in Apache Atlas (Tag Sync), and plugins for components of the Hadoop ecosystem.

Plugins are lightweight Java programs installed on components of Hadoop (Hive Server, HDFS Name Node, and YARN Resource Manager) to periodically pull policies from the central Ranger server and store them locally using a REST API. When a user request comes to any Hadoop component, the plugin intercepts it and evaluates it against the security policy to decide whether the user request should be authorized.

The advantages of using Apache Ranger with Hadoop are primarily associated with centralizing the security administration of many security settings and tasks through a single WebUI. Some areas, such as OS security settings, are important for cluster security but are not managed by Ranger. This helps provide consistent AC across all Hadoop components through Ranger plugins. The use of User Sync and Tag Sync is instrumental to fine-grained AC and facilitates ABAC-style security.

Apache Atlas

Apache Atlas provides a framework for managing metadata associated with data and processes. Software program hooks are installed on components of the ecosystem, e.g., the HDFS Name Node and YARN Resource Manager, to communicate information on the data to Atlas so that the metadata can be organized and stored based on the given relationships and types. These metadata can then be used by Ranger to define and enforce tag-based security policies. The definition and management of the metadata are necessary for tracking data provenance.

Atlas is designed to be extensible. Thus, if a hook does not exist for a new component of the Hadoop ecosystem, it can be developed. Hooks are lightweight programs (e.g., script cURL commands) written to interface the tracked object types (e.g., data, databases, and programs) to the repository of Atlas metadata through a REST API.

In addition to using pre-defined types, new metadata types can be defined, including complex types that inherit attributes from other types of metadata. The metadata can be classified with attributes to indicate their sensitivity, e.g., PII, and the classification can be propagated from metadata on the process input to those on the output. The lineage of an entity, an instance of a type of metadata, can be tracked by using this feature of propagation as the data are processed through a lifecycle.

When we install Apache Atlas, we select H-Base to store the metadata and Solar to store the index, where this is the default configuration. Other core components include the graph engine that supports the relationships between entities. For example, the relations of columns and databases and files and folders are managed by the graph engine. The ingest/export features capture messages from the sources of metadata, create entities, and update events.

A challenge in Apache Atlas is that nothing is performed automatically. One needs to launch the hooks, i.e., cURL commands, in scripts that run regularly on the component systems to crawl through the data store. Some hooks are available from the Apache Atlas repository for installation and configuration on the core components. For other data sources, one has to write hooks by using the cURL calls to register and tag the entities.

In Figure 8, we show a simple entity (instance) of an HDFS file type (class) created in our experiment with a small number of attributes used as a basis for policies. Additional more complex examples of JSON files used to create entities and types in Atlas are

posted at our open data repository (https://github.com/AnneMT/SEHadoop/tree/main/SEHadoop/atlas/atlas-files accessed 14 January 2023).

```
{
"entity": {
"typeName": "hdfs_folder",
"attributes": {
"owner": "hdfs",
"createTime": 1619297668000,
"qualifiedName": "DataSet1_patientsDFData@sehadoop",
"displayName": "patientsDFData folder",
"name": "patientsDFData",
"description": "patients data filtered and in DataFrame format"
}
}
}
```

Figure 8. Example Atlas entity for HDFS folders with attributes.

3. Results

To analyze ABAC security in BDP, we installed and configured Apache Hadoop on AWS EC2 instances. We have posted our detailed system configuration, programs, and data on GitHub (https://github.com/AnneMT/SEHadoop accessed 14 January 2023) to allow other users to replicate our results and continue developing security-related insights. The subsequent sections provide an overview of the configuration of the experiment as well as our analysis and recommendations. The configuration is shown in Figure 9.

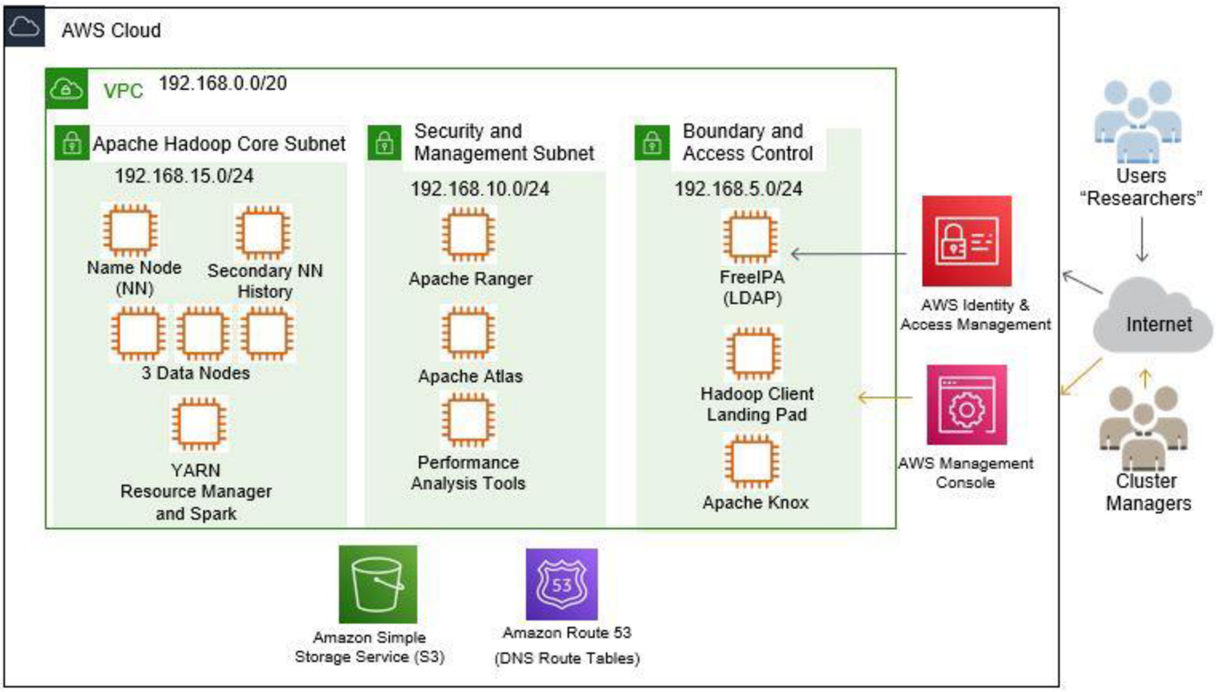

Figure 9. AWS EC2 instances for the security analysis of Apache Hadoop.

3.1. Software and AWS EC2 Instances

The BDP representation for the experiment was built on AWS EC2 instances by using the CentOS 7 Amazon Machine Image (AMI). The following open-source Apache software

was included: Hadoop version 2.7.3, Spark version 3.1.2, Ranger version 2.1.0, and Atlas version 2.1.0. We used seven EC2 instances to install the Hadoop Name Node, secondary Name Node, YARN Resource Manager with Spark, and three Data Nodes. Two more instances were used for Apache Ranger and Atlas. FreeIPA was used for directory services (i.e., LDAP) on another EC2 instance for a total of 10 EC2 instances. This configuration allowed us to isolate potential performance-related issues and independently manage the configurations of components of the Hadoop ecosystem. Additional EC2 instances were configured to run the system management tools and the boundary security services of Apache Knox. Although we did not fully investigate the security services of the boundary proxy server of Apache Knox, it was identified as an important part of a layered security architecture.

3.2. Datasets

Two synthetically generated datasets were incorporated into the experiment: data on electronic healthcare records (EHR) generated by Synthea™ (https://github.com/synthetichealth/synthea/wiki accessed 14 January 2023) and representative messages from Twitter generated by SynSocial (https://github.com/AnneMT/SynSocial accessed 14 January 2023). This combined dataset realistically represented a connected dataset with multiple sensitivities. Synthea generated HL7 files that represented the medical histories of patients, and SynSocial generated Twitter messages based on the medical conditions in the HL7 patient files. We thus created a large dataset with varying levels of sensitive information such that challenges to BDP-specific performance could be analyzed [54]. Our dataset represented 1000 users/patients, had a size of 14 GB and was configured with the block replication set to three.

3.3. Data Processing Lifecycle

Apache Spark programs were developed to process the data in a representative data processing lifecycle, as depicted in Figure 10. The BDP lifecycle represented consisted of six PySpark programs that processed the data sequentially. Starting with two datasets from synthetic data generators, the data were filtered, joined, and protected with a privacy-preserving hash and encryption and were finally analyzed to output the aggregated results. This data processing sequence represents a realistic yet simplified set of steps that require security for the data. We executed the programs with different levels of permission checking for users. The process generated data with attribute metadata tags that reflected changes in the security classification of the data (sensitivity). We started with two datasets and different levels and generated datasets with lower sensitivity such that the final set contained hashed or encrypted personally identifiable information (PII) as well as data from the Heath Insurance Portability and Accountability Act (HIPAA).

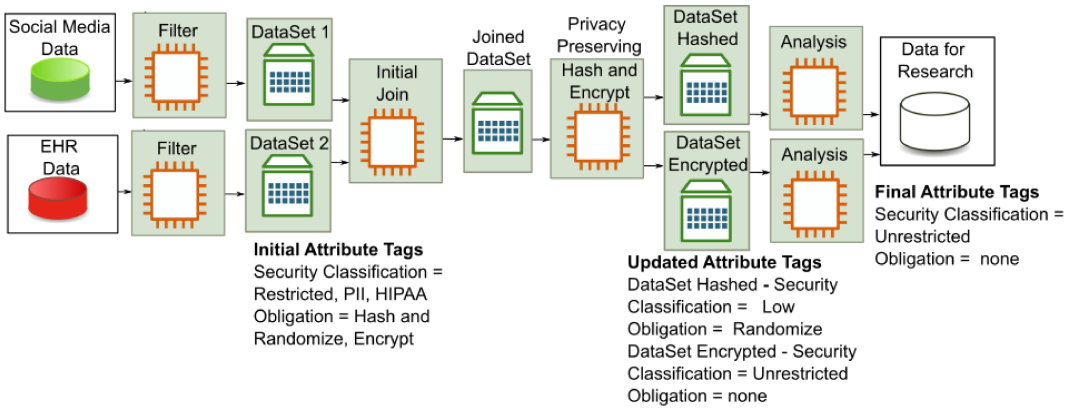

Figure 10. Lifecycle of data processing for security experiments.

3.4. Analysis and Observations

The major challenge for distributed data processing is that the security-related issues that previously plagued a computer system now occur across a network of computers with applications and data in multiple locations. In addition to observations of AC security services for the components described above, we observed several issues in the system-wide approach to security. We also captured performance-related information under different security configurations and found negligible impact. Apache Hadoop delivers impressive performance, and further security enhancements will increase its popularity. Application of our framework to the design and configuration will enable multi-tenant and multi-sensitivity data analysis.

3.4.1. ABAC Support

The primary goal of our experiment was to examine the ability to support fine-grained ABAC. The results indicate that this is feasible, but this requires significant software configurations, including writing scripts to connect the metadata to the policy enforcement program. There was a high risk of incorrect configuration.

ABAC could be achieved by using policies of Apache Ranger that bring together attributes from three sources, as depicted in Figure 11. Such user information as group assignments could be provided by the directory service over LDAP by using Apache Ranger's User Sync. Ranger plugins for the HDFS and YARN were used to enable the configuring of policies on file permissions and job scheduling (e.g., capacity scheduler queues). The Tag Sync capability of Atlas allowed the metadata for the HDFS and YARN to be used as a basis for these security policies. Thus, for example, data could be tagged or classified as PII or HIPAA in Atlas, and these tags were then used as the basis for security policies that allowed certain users or groups to access the files. YARN policies could likewise be configured to permit or deny users access to submit jobs to queues based on tags from Atlas. This same strategy could be used with Apache Hive and other components. The classification tags from Atlas and policies created in Ranger provided the capabilities to achieve ABAC, but there were limitations associated with flexibility in assigning attributes to users. Users could be managed locally in Ranger or externally through a directory service and assigned to groups that could be mapped to the concept of attributes for policy-related decisions. Assigning users to multiple groups and then using these groups as part of the process of policy decision provided the concept of attribute-based control, but this might not have provided the flexibility of key–value pair attribute-based controls described in the reference implementations [55].

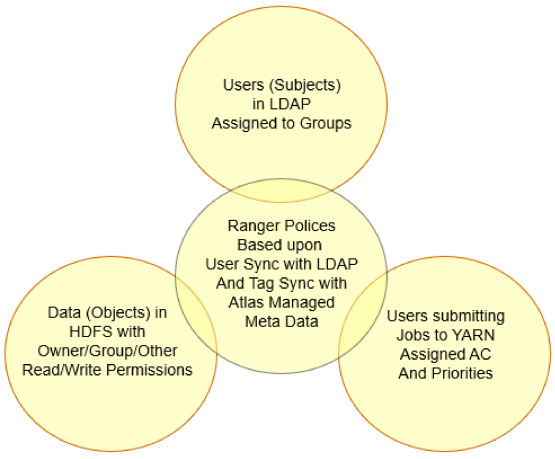

Figure 11. Implementation of ABAC using the HDFS, LDAP, YARN, Ranger, and Atlas.

3.4.2. Multiple Management Consoles

Although several management frameworks are available for Apache Hadoop, such as Apache Ambari and the Cloudera Manager, a single management console that fully integrates across the ecosystem is not available. Interfaces for system security management, such as the Apache Ranger Admin WebUI, are separate from other system management consoles, such as Apache Ambari.

A limitation of the Apache Ambari and Cloudera Manager is that they must be used to set up the cluster that they manage. One cannot add these managers to clusters that are already set up. These tools are designed to tightly couple with their Hadoop configuration.

From a security perspective, these management tools help configure or "turn on" security components but do not confirm whether the system is fully secure. Validating the enforcement of the security policy requires interfacing to separate consoles, including directory systems and the Ranger Admin WebUI. Confirming that the system is correctly configured and reviewing its security status requires examining multiple interfaces and files. Automating the collection of this status-related information requires significant development of custom scripts that push logs to a central location.

3.4.3. Verification of Security Software

Software that provides security functionality should be subject to rigorous and independent evaluation. An example of this is the Common Criteria Evaluated Assurance Levels and Target Profiles of Protection, applied to operating systems to test their security services. Governments and businesses prefer the Red Hat version of Linux because of its independent certification (https://www.redhat.com/en/about/press-releases/red-hat-adds-common-criteria-certification-red-hat-enterprise-linux-8 accessed 14 January 2023).

Independently writing security software and hooks to interface with critical security services, such as the Kerberos Key Distribution Center (KDC), can help circumvent this type of rigorous testing. Exposure to flawed error handling, buffer overflows, brute force attacks and poor audit logging are all areas that require independent verification [56].

Veracode's State of Software Security Report found that more than three-quarters (75.2%) of applications have security flaws [57]. It is a large software manufacturer that did not include small glue code rapidly developed under business pressures to obtain investment in BDP.

In the analysis, we noted several security issues in the software components. Many of the configuration files contained administrator passwords in clear text. Log files, which can contain sensitive information such as Kerberos service tickets, were stored in multiple potentially unprotected locations. 3DES and R4 are the data encryption algorithms used in Apache Hadoop and are relatively weak [58]. Additional configuration is required to incorporate the AES. Rigorous security testing can help identify design flaws that can then be remedied.

3.4.4. Performance

We analyzed the performance of Apache Hadoop under different security configurations. We measured the elapsed execution time for each sequential job. The workflow of data inputs, PySpark programs, and outputs was assigned attributes and tags by using Apache Atlas, as shown in Figure 12.

The elapsed execution time was recorded from the Hadoop Resource Manager's web GUI under different security configurations, as listed in Table 4 and shown in Figure 13. The configurations were as follows: execution with the Ranger plugins disabled, execution as a local privileged user of Hadoop, execution with resource-based AC by using policies of Apache Ranger, and execution with tag-based AC that uses both the polices of Apache Ranger and the metadata of Apache Atlas. The fourth configuration corresponds to an ABAC approach. The performance analysis did not yield any significant differences in terms of executing the sequence of data processing programs (PySpark programs) in the use case for the lifecycle of the data. Overall, we noted similar program execution times in

all three configurations. This indicates that AC security services had a very limited impact on the performance of the system.

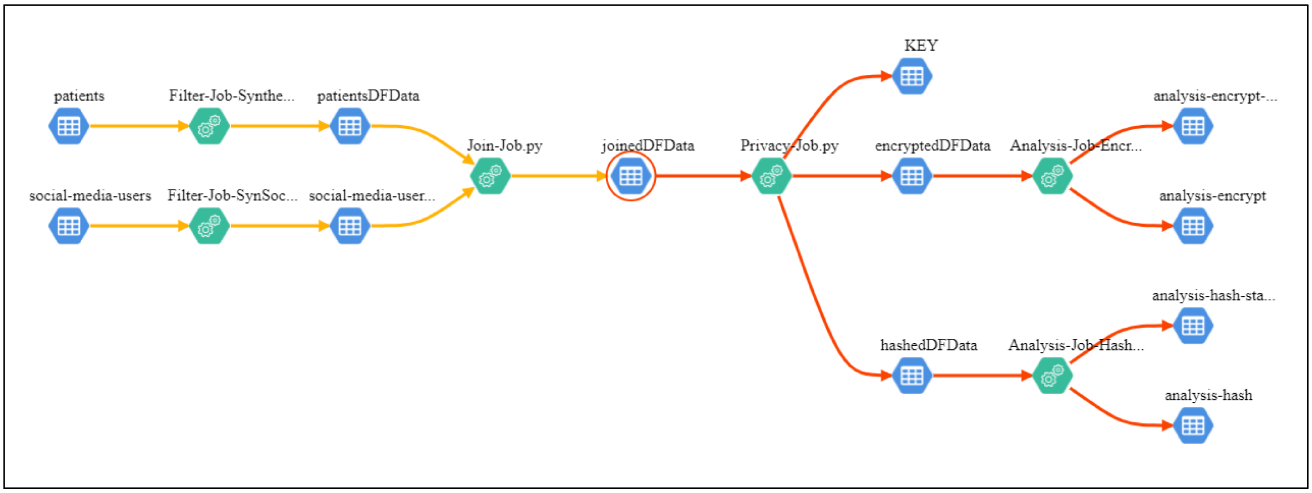

Figure 12. Configuration of data processing and propagation of attribute classification in Atlas.

Table 4. Impact of AC security on program execution time.

PySpark Program	Description	Elapsed Times (s) (Average of 15 Executions)			
		Ranger Plugins Disabled	Local Privileged User	Resource-Based AC	Tag-Based AC
1—Filter-Job-SynSocial.py	Filters JSON data from the Twitter dataset and puts it into Spark DataFrames	62.50	61.20	61.93	64.00
2—Filter-Job-Synthea.py	Filters HL7 JSON data from a healthcare dataset and puts it into Spark DataFrames	142.73	139.13	133.93	129.67
3—Join-Job.py	Joins the social media and healthcare datasets based on the social media messages and healthcare condition for each patient	53.47	50.87	53.07	53.07
4—Privacy-Job.py	Hashes and encrypts PII and HIPAA data	334.80	357.27	330.80	317.33
5—Analysis-Job-Hash.py	Analyzes hashed data from the private job to produce the number of messages per medical condition	58.00	60.33	61.07	62.73
6—Analysis-Job-Encrypt.py	Analyzes the encrypted data from the private job to generate the number of messages per medical condition	95.20	95.20	91.67	89.40

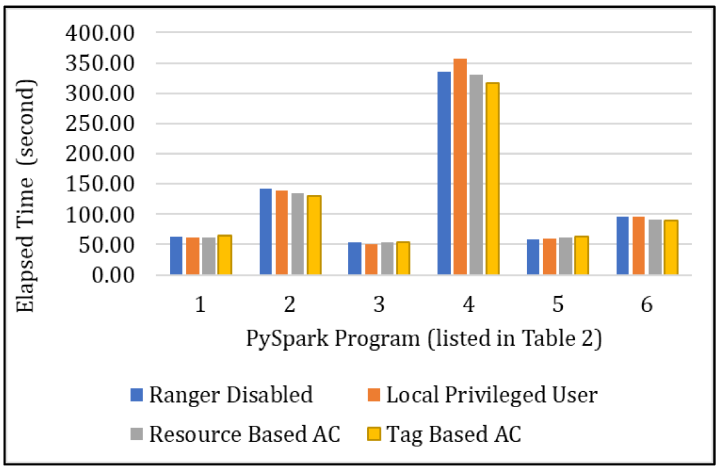

Figure 13. Elapsed times for executions of the PySpark program under different configurations.

Although at least minor differences in performance were expected when executing the lifecycle of the data under different security configurations, we needed to scale out significantly to observe such impacts on performance according to performance analyses reported in the industry (https://privacera.com/blog/immuta-vs-apache-ranger-a-failed-benchmark/ accessed 14 January 2023). We estimated that we needed to scale up from 14 GB to over 1 TB of data, which would require six or more Data Nodes for the observation of a minor (under 10%) impact on performance.

As the size of the dataset and the complexity of the processing lifecycle were scaled out, the impact of performance could have been identified by the increase in the elapsed time. However, the overhead due to AC services provided by Apache Ranger was expected to be minimal. The encryption of the data and Kerberos led to a more significant increase in the elapsed time.

We considered the performance analysis that was conducted by other researchers, such as Gupta et al. [59]. In their experiment, a remote ABAC policy enforcement approach was used with MongoDB (another open-source large data set framework), and performance impacts were not detected until they scaled out to over 500 policies and 20 attributes per entity. This large number of policies seems impractical in that it would become extremely difficult to manage. Most operational implementations would have far fewer policies and attributes per entity. Using the Ranger and Atlas framework for ABAC, the overhead processing impact is additionally minimized because policies are enforced by a local Ranger plugin API rather than remotely. Therefore, we believe that the performance of our approach would be significantly better and not impact a parallel processing environment with large datasets. We did not detect a significant overhead in scaling out the Tag-based AC approach with Ranger and Atlas. However, additional experiments with increasingly larger data sets and complex programs could identify upper limits.

The Transaction Processing Performance Council (TPC) has several benchmarks for big dataset analysis for Hadoop ecosystems. The TPC Benchmark™HS (TPCx-HS) benchmark workload is based on the Hadoop TeraSort program, for example (https://www.tpc.org/tpcx-hs/ accessed 14 January 2023). These benchmarks provide insights into acquiring hardware for a Hadoop cluster but do not address the security-specific impacts on performance, such as encryption.

Additional tools are available to monitor the performance of a large-scale Hadoop cluster. The AWS EC2 interface also provides statistics on the processor, memory, storage, and network bandwidth usage. Several server- and network-monitoring tools, although not specific to Hadoop, are open-source, or a free version of them is available for use, such as Ganglia (http://ganglia.sourceforge.net/ accessed 14 January 2023) and Nagios (https://exchange.nagios.org/ accessed 14 January 2023). A feature of many of these management tools is that the areas monitored and the reports generated are customizable. Areas that are a focus for troubleshooting can be displayed on a GUI. Hadoop-specific tools, such as Apache Ambari and Cloudera Manager, must be used to install all components of the Hadoop ecosystems and cannot be added to an installed cluster. These tools provide Hadoop-specific statistics, such as the status of data block replication and YARN job schedule management.

3.5. Limitations

The experiment we conducted had some limitations. Specifically, because our configuration used open-source software, we were not able to discover to what extent default configurations of commercial BDP software address our security findings. These types of commercial configurations most likely would include more status monitoring and logging capabilities to detect misconfigurations and security issues. However, these types of management interfaces could also obscure security issues in the system-to-system cluster computing communication links. Also, the extent to which standards are met to enable the interoperability of security features in independently developed systems was not thoroughly examined.

4. Discussion

Our approach to BDP security, i.e., our recommended framework, includes implementation considerations as the third area. Specifically, the following areas should be advanced and incorporated into the design and implementation of a secure BDP configuration:

1. Exclusive use of a data analyst notebook that provides a secure interface cluster,
2. A data security service layer that is integrated into all components of the BDP system.
3. A complete central management system that provides a single console to show the status of all components of the system.

Without these fundamental controls, more advanced security services such as ABAC will be difficult to achieve in a complete and integrated manner. Table 5 below summarizes the approach to these implementation requirements. These enhancements help to address Common Weaknesses and Exploits (CWE) (https://cwe.mitre.org/ accessed 14 January 2023). We believe these can largely be based upon integrating and expanding current capabilities. Therefore, we rate the development difficulty as a medium. Notebook development would be easier based on previous work, and the security service layer would be harder due to the complexity and diversity of the Hadoop ecosystem.

Table 5. Implementation considerations were identified in the prototype experiment.

Recommended Enhancement	Weaknesses Addressed	Potential Development Strategy	Development Difficulty
Secure Data Analysts Notebook	Improper Input, SQL Injection (CWE-79 and 89)	Develop from Apache HUE, Zeppelin or Jupyter Notebooks	Medium-Easy
Data Security Service Layer	Improper Authentication (CWE-287) Missing Authorizations (CWE-862)	Build upon Ranger plugins and Hadoop Secure Mode capabilities (in later versions)	Medium-Hard
Management Console	Improper Restriction of Operations (CWE-119) Concurrent Execution, Race Condition (CWE-362)	Integrate Hadoop Ambari, Ranger, and computer-network security-performance analysis tools	Medium

A commercial and managed distribution, which addresses these areas, is required if Apache Hadoop is used in business applications. This should be used to provide the add-ons and integration glue code necessary for logging, security services, and command and control of the management center. A professional, team-based approach is needed to handle the increasingly complex environment when larger and more diverse tools are used to meet the requirements of analysis. System security managers and administrators face significant challenges in understanding and addressing the security of all the different ports, protocols, and services that are externally accessible within each component.

4.1. Secure Data Analyst Notebook

An easy-to-use, consistent, and secure tool that provides the sole data analyst interface for the cluster is needed. This can help ensure that the security settings are consistently enforced by users with multiple authorizations. Examples of currently used data analyst notebooks include Apache HUE and Zeppelin. These notebooks allow flexibility in executing a variety of searches and analytical programs by using the selected ecosystem tools. Areas for enhancement in this domain include ensuring that the tools of analysis are under configuration control and that the security policies are enforced in a verifiable manner. A challenge in current configurations of the Hadoop ecosystem, such as Apache Atlas, is a lack of security enforcement.

Tools that can be configured to use SSL, HTTPS over REST API interfaces to protect data in transit, and Kerberos and LDAP for authentication and authorizations should be used. For example, Jupyter notebooks depend on a gateway-enabled notebook server, such as Apache Knox or Spark Magic with Livy, to support Kerberos. User impersonation

is used to submit jobs to the cluster, resulting in complicated individual accountability. Unintentional damage can occur when users can take on the credentials of the superuser.

4.2. Data Security Service Layer

The Hadoop ecosystem is based on a group of independently developed projects that are connected through open protocols rather than built from the ground up with an integrated security service layer. This results in the wide use of access through accounts with root-level privileges to overcome software-related problems. A significant challenge in setting up a BDP ecosystem is that all the software components have different schedules of version updates, and not all versions are compatible with one another. Enforcing a common security service layer that needs to be invoked to interoperate would help ensure that security services are not disabled or bypassed as software versions change.

The status of the security features of Apache Hadoop is that they are added on rather than considered as invoked security services that are fully integrated through standard interfaces as part of compliance with ecosystem integration. Examples of security service layers include the Oracle Platform Security Services (https://docs.oracle.com/cd/E21764_01/core.1111/e10043/underjps.htm#JISEC1827 accessed 14 January 2023) for data processing systems and the Generic Security Standard Application Programming Interface (GSS-API) for software development (https://docs.oracle.com/cd/E19683-01/816-1331/overview-6/index.html accessed 14 January 2023).

Apache Ranger is a correct step in this direction, but the developers' goals of flexibility and extensibility can undermine consistency in security services for components. Each capability can be optionally implemented to result in fractured security. Apache Ranger needs to be developed further into a complete and integrated central security service that applies standard interfaces, such as XACML. Such an integrated and comprehensive security service can provide an underlying plan that is invoked consistently by all BDP system services.

4.3. Perspective of Central Management

Given the importance of BDP, system security managers and administrators need to think and operate similar to cybersecurity attackers. A perspective of central management is needed that supports this philosophy and provides complete information on the status of security. This would be the result of active and continuous testing of each component of the ecosystem as well as the collective environment. Such testing needs to be carried out across the cybersecurity lifecycle. Penetration scanning that attempts to exploit open ports, protocols, and services to gain a foothold in the system can be provided by using such tools as Nmap and OpenVAS. The exposure to exploits that exfiltrate or corrupt data by establishing command and control through campaign kits such as Metasploit needs to be well understood. Any SQL input should be subject to SQL injection testing in a manner that informs the security team.

Information on audit logs needs to be centrally accessible so that behaviors can be correlated together and correspond to user and process actions. A much stronger confirmation of policy enforcement can be provided when the results of log analysis are pulled from all components of the ecosystem, including the underlying operating systems and the components of network security.

Traffic analysis by using, for example, firewalls and intrusion detection should also be available to the security manager so that attempts to maintain or reestablish a presence (of unauthorized software) on BDP systems can be detected. On the whole, the principle of least privilege needs to be applied by ensuring individual accountability and restricting the shared use of superuser accounts.

5. Conclusions

With the availability of next-generation BDP tools such as Apache Hadoop and cloud-based Hadoop-like ecosystems, organizations are beginning to recognize the potential

for the storage and processing of larger volumes of different types of data. This includes sensitive data that need to be protected not only to meet compliance regulations but also to protect investments in high-integrity data that inform business decisions. Our analysis of the security of Apache Hadoop here, which focused on ABAC, identified an approach to analyzing vulnerabilities and potential attacks. We identified several security issues based on our experimental environment. We have also made our results available to support additional related research. An overall security goal of BDP is to achieve fine-grained AC through ABAC. In this analysis, we defined a framework that provides a strong foundation to achieve BDP security. This framework consists of three key actions: analysis of standards and fundamental reference architectures, realistic prototype testing, and implementation considerations. XACML and NGAC were identified as the standard references for ABAC. By using components of the open-source Apache Hadoop ecosystem, we analyzed the security-related features in detail. Commercial versions can hide the security features in compiled code. A detailed analysis of potential vulnerabilities supports rigorous testing and new innovative approaches to security. Finally, we identified three main areas that need to be pursued to attain operational security: the mandatory use of a secure data analysts' notebook, a data security service layer, and a central management console. These capabilities are partially met today and require enhancements through continued research on BDP security to attain their ultimate objective.

Author Contributions: Both authors contributed substantially to this research work and paper reporting the results. All authors have read and agreed to the published version of the manuscript.

Funding: This research was sponsored by the U.S. National Science Foundation (NSF) under Grant 1915780.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data generator used to create a representative healthcare data set is available at: https://github.com/synthetichealth/synthea/wiki accessed 14 January 2023; the data generator used to create social media data based upon the healthcare dataset is available at: https://github.com/AnneMT/SynSocial accessed 14 January 2023. Example datasets, PySpark programs, data generation tools, and details on Hadoop, Ranger, and Atlas configurations are available at https://github.com/AnneMT/SEHadoop accessed 14 January 2023.

Acknowledgments: MITRE Approved for Public Release; Distribution Unlimited. Public Release Case Number 22-0810 and 22-1549. The author's affiliation with The MITRE Corporation is provided for identification purposes only and is not intended to convey or imply MITRE's concurrence with or support for the positions, opinions, or viewpoints expressed by the author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chandrakar, I.; Hulipalled, V.R. Privacy Preserving Big Data mining using Pseudonymization and Homomorphic Encryption. In Proceedings of the 2021 2nd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 1–3 October 2021.
2. Sellami, R.; Zalila, F.; Nuttinck, A.; Dupont, S.; Deprez, J.-C.; Mouton, S. FADI–A Deployment Framework for Big Data Management and Analytics. In Proceedings of the 2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Bayonne, France, 10–13 September 2020.
3. Colombo, P.; Ferrari, E. Access Control in the Era of Big Data: State of the Art and Research Directions. In Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies (SACMAT'18), Indianapolis, IN, USA, 13–15 June 2018.
4. Horton, N.; DeSimone, A. *Sony's Nightmare before Christmas: The 2014 North Korean Cyber Attack on Sony and Lessons for US Government Actions in Cyberspace*; Defense Technical Information Center: Laurel, MD, USA, 2018.
5. Saleem, H.; Naveed, M. SoK: Anatomy of data breaches. *Proc. Priv. Enhancing Technol.* **2020**, *4*, 53–174. [CrossRef]
6. Hart, M. *Kerberos Attacks: What You Need to Know*; Cyberark: Newton, MA, USA, 2015.
7. George, L. *User Name Handling in Hadoop*; OpenCore: Wedel, Germany, 2016.
8. Hu, V.; Ferraiolo, D.; Kunn, R.; Schnitzer, A.; Sandlin, K.; Miller, R.; Scarfone, K. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations SP 800-162*; NIST: Gaithersburg, MD, USA, 2014.

9. International National Committee for Information Technology Standards (INCITS). *Information Technology—Next Generation Access Control—Implementation Requirements, Protocols and API Definitions (NGAC-IRPAD)*; InterNational Committee for Information Technology Standards: Washington, DC, USA, 2018; Volume 525, pp. 1–44.
10. Sen, S.; Guha, S.; Datta, A.; Rajamani, S.; Tsai, J.; Wing, J. Bootstrapping Privacy Compliance in Big Data Systems. In Proceedings of the 2014 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 18–21 May 2014.
11. Zhioua, Z.; Ameur-Boulifa, R. Framework for the Formal Specification and Verification of Security Guidelines. *Adv. Sci. Technol. Eng. Syst. (ASTES) J.* **2018**, *3*, 38–48. [CrossRef]
12. Hu, V.; Ferraiolo, D.; Kuhn, R. *Attribute Considerations for Access Control Systems*; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2019.
13. Nguyen, D. Provenance-Based Access Control Models. Ph.D Thesis, Department of Computer Science, University of Texas at San Antonio, San Antonio, TX, USA, 2014.
14. Liao, C.; Squicciarini, A. Towards Provenance-Based Anomaly Detection in MapReduce. In Proceedings of the IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID), Shenzhen, China, 4–7 May 2015; IEEE: Shenzhen, China, 2015.
15. Sun, L.; Park, J.; Nguyen, D.; Sandhu, R. A Provenance-Aware Access Control Framework with Typed Provenance. *IEEE Trans. Dependable Secur. Comput.* **2016**, *13*, 411–423. [CrossRef]
16. Won, H.; Nguyen, M.C.; Gil, M.-S.; Moon, Y.-S. Advanced Resource Management with Access Control for Multitenant Hadoop. *J. Commun. Netw.* **2015**, *17*, 592–601. [CrossRef]
17. Solanki, N.; Huang, Y.; Yen, I.-L.; Bastani, F.; Zhan, Y. Resource and Role Hierarchy Based Access Control for Resourceful Systems. In Proceedings of the 2018 IEEE 42nd Annual Computer Softwae and Applications Conference (COMPSAC), Tokyo, Japan, 23–27 July 2018.
18. Yu, Y.; Chen, Y.; Wen, Y. Task-role based access control model in logistics management system. In Proceedings of the 2013 IEEE International Conference on Service Operations and Logistics, and Informatics, Dongguan, China, 28–30 July 2013.
19. Alshammari, S.; Albeshri, A.; Alsubhi, K. Integrating a High-Reliability Multicriteria Trust Evaluation Model with Task Role-Based Access Control for Cloud Services. *Symmetry* **2021**, *3*, 492. [CrossRef]
20. Wang, P.; Jiang, L. Task-role-based Access Control Model in Smart Health-care System. In Proceedings of the MATEC Web of Conferences International Conference on Engineering Technology and Application (ICETA 2015), Nagoya, Japan, 29–30 May 2015.
21. Ma, L.; Tao, L.; Zhong, Y.; Gai, K. RuleSN: Research and Application of Social Network Access Control Model. In Proceedings of the 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), New York, NY, USA, 9–10 April 2016.
22. Cheng, Y.; Park, J.; Sandhu, R. An Access Control Model for Online Social Networks Using User-to-User Relationships. *IEEE Trans. Dependable Secur. Comput.* **2016**, *13*, 424–436. [CrossRef]
23. Rizvi, S.Z.R.; Fong, P.W.; Crampton, J.; Sellwood, J. Relationship-Based Access Control for an Open-Source Medical Records System. In Proceedings of the SACMAT'15: 20th ACM Symposium on Access Control Models and Technologies, Vienna, Austria, 1–3 June 2015.
24. Ma, L.; Tao, L.; Gai, K.; Zhong, Y. A novel social network access control model using logical authorization language in cloud computing. *Concurr. Comput. Pract. Exp.* **2016**, *9*, 1–17. [CrossRef]
25. Zhang, R.; Liu, L.; Xue, R. Role-based and time-bound access and management of EHR data. *Secur. Commun. Netw.* **2014**, *7*, 994–1015. [CrossRef]
26. Yang, K.; Liu, Z.; Jia, X.; Shen, X.S. Time-Domain Attribute-Based Access Control for Cloud-Based Video Content Sharing: A Cryptographic Approach. *IEEE Trans. Multimed.* **2016**, *18*, 940–950. [CrossRef]
27. Gupta, M.; Patwa, F.; Sandhu, R. Object-Tagged RBAC Model for the Hadoop Ecosystem. In Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy DBSEC 2017, Philadelphia, PA, USA, 19–21 July 2017.
28. Kayes, A.; Han, J.; Colman, A. An ontological framework for situation-aware access control of software services. *Inf. Syst.* **2015**, *53*, 253–277. [CrossRef]
29. Kumar, T.A.; Liu, H.; Thomas, J.P.; Hou, X. Content sensitivity based access control framework for Hadoop. *Digit. Commun. Netw.* **2017**, *3*, 213–225. [CrossRef]
30. Zeng, W.; Yang, Y.; Luo, B. Access control for big data using data content. In Proceedings of the 2013 IEEE International Conference on Big Data, Silicon Valley, CA, USA, 27 June–2 July 2013.
31. Morgan, R.L.; Cantor, S.; Carmody, S.; Hoehn, W.; Klingenstein, K. Federated Security: The Shibboleth Approach. *EDUCASE Q.* **2004**, *27*, 12–17.
32. OASIS. *Cross-Enterprise Security and Privacy Authorization (XSPA) Profile of SAML v2.0 for Healthcare, Version 2.0, Committee Specification 01*; OASIS: Burlington, MA, USA, 2019.
33. HL7 International. *HL7 Healthcare Privacy and Security Classification System (HCS), Release 1, Ann*; HL7 International: Arbor, MI, USA, 2014.
34. Fu, X.; Nie, X.; Wu, T.; Li, F. Large universe attribute based access control with efficient decryption in cloud storage system. *J. Syst. Softw.* **2018**, *135*, 157–164. [CrossRef]

35. Li, J.; Zhang, Y.; Ning, J.; Huang, X.; Poh, G.S.; Wang, D. Attribute Based Encryption with Privacy Protection and Accountability for CloudIoT. *IEEE Trans. Cloud Comput. (Early Access)* **2020**, *10*, 762–773. [CrossRef]
36. Teng, W.; Yang, G.; Xiang, Y.; Zhang, T.; Wang, D. Attribute-Based Access Control with Constant-Size Ciphertext in Cloud Computing. *IEEE Trans. Cloud Comput.* **2017**, *5*, 617–627. [CrossRef]
37. Wang, J.; Crawl, D.; Purawat, S.; Nguyen, M.; Altintas, I. Big data provenance: Challenges, state of the art and opportunities. In Proceedings of the 2015 IEEE International Conference on Big Data, Santa Clara, CA, USA, 29 October–1 November 2015.
38. Hellerstein, J.; Sreekanti, V.; Gonzalez, J.; Dalton, J.; Dey, A.; Nag, S.; Ramachandran, K.; Arora, S.; Bhattacharyya, A.; Das, S.; et al. A Data Context Service. In Proceedings of the CIDR 2017, Chaminade, CA, USA, 8–11 January 2017.
39. Sowmy, Y.; Nagaratna, M.; Shoba Bindu, C. M-SANIT: A Framework for Effective Big Data. *J. Theor. Appl. Inf. Technol.* **2018**, *96*, 1596–1605.
40. Nagajothi, S.; Raj Kumar, N. Data Anonymization Technique for Privacy Preservation Using MapReduce Framework. *Int. J. Adv. Res. Comput. Commun. Eng.* **2016**, *5*, 1012–1018.
41. Zhang, X.; Yang, L.; Liu, C.; Chen, J. A Scalable Two-Phase Top-Down Specialization Approach for Data Anonymization Using MapReduce on Cloud. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 363–373. [CrossRef]
42. Saraladevi, B.; Pazhaniraja, N.; Victer Paul, P.; Saleem Basha, M.; Dhavachelvan, P. Big Data and Hadoop-a Study in Security Perspective. *Procedia Comput. Sci.* **2015**, *50*, 596–601. [CrossRef]
43. Cloud Security Alliance; Top Threats Working Group. *Top threats to Cloud Computing: Egregious Eleven*; Cloud Security Alliance: Seattle, WA, USA, 2019.
44. Khandelwal, S. *Insecure Hadoop Clusters Expose Over 5,000 Terabytes of Data*; The Hacker News: New York, NY, USA, 2017.
45. Bhathal, G.S.; Singh, A. Big Data: Hadoop framework vulnerabilities, security issues and attacks. *Array* **2019**, *1–2*, 1–8. [CrossRef]
46. Fu, X.; Gao, Y.; Luo, B.; Du, X.; Guizani, M. Security Threats to Hadoop: Data Leakage Attacks and Investigation. *IEEE Netw.* **2017**, *31*, 67–71. [CrossRef]
47. Mondal, P. *Thousands of Unauthenticated Databases Exposed on the Internet*; RedHunt Labs: London, UK, 2021.
48. Kolesnikov, O.; Parashar, H. *Detecting Persistent Cloud Infrastructure/Hadoop/YARN Attacks Using Security Analytics: Moanacroner, XBash, and Others*; Securonix Threat Research: Jersey City, NJ, USA, 2019.
49. Sinha, S.; Gupta, S.; Kumar, A. Emerging Data Security Solutions in HADOOP based Systems: Vulnerabilities and Their Countermeasures. In Proceedings of the 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 18–19 October 2019.
50. Cheng, L.; Shen, Q.; Dong, C. *Invader Job: A Kind of Malicious Failure Job on Hadoop YARN*; IEEE: Kansas City, MO, USA, 2018.
51. Geenens, P. *Hadoop YARN: An Assessment of the Attack Surface and Its Exploits*; Radware: Mahwah, NJ, USA, 2018.
52. Antony, B. *Secure Communication in Hadoop without Hurting Performance*; Ebay: San Jose, CA, USA, 2016.
53. Bhamidimarri, V.R. *Introducing Amazon EMR Integration with Apache Ranger*; AWS: Washington, DC, USA, 2021.
54. Tall, A.; Zou, C.; Wang, J. Generating Connected Synthetic Electronic Health Records and Social Media Data for Modeling and Simulation. In Proceedings of the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC), Orlando, FL, USA, 1–3 December 2020.
55. Bhatt, S.; Patwa, F.; Sandhu, R. ABAC with Group Attributes and Attribute Hierarchies Utilizing the Policy Machine. In Proceedings of the 2nd ACM Workshop on Attribute-Based Access Control (ABAC'17), New York, NY, USA, 24 March 2017.
56. IEEE Computer Society Center for Secure Design. *Avoiding the Top 10 Software Security Design Flaws*; IEEE: Washington, DC, USA, 2015.
57. Veracode. *State of Software Security Volume 11*; Veracode: Burlington, MA, USA, 2020.
58. Kapil, G.; Agrawal, A.; Attaallah, A.; Algarni, A.; Kumar, R.; Kahn, R.A. Attribute based honey encryption algorithm for securing big data: Hadoop distributed file system perspective. *PeerJ Comput. Sci.* **2020**, *6*, e259. [CrossRef] [PubMed]
59. Gupta, E.; Sural, S.; Vaidya, J.; Atluri, V. Enabling Attribute-based Access Control in NoSQL Databases. *IEEE Trans. Emerg. Top. Comput.* **2022**, 1–15. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.