BIG DATA PROCESSING
ATTRIBUTE BASED ACCESS CONTROL SECURITY

by

ANNE M. TALL
B.S. Electrical Engineering University of Maryland, 1987
Masters Electrical Engineering Johns Hopkins University, 1993

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Engineering and Computer Science
in the College of Computer Engineering
at the University of Central Florida
Orlando, Florida

Spring Term
2022

Major Professors:
Changchun Zou
Jun Wang

ABSTRACT

The purpose of this research is to analyze the security of next-generation big data processing (BDP) and examine the feasibility of applying advanced security features to meet the needs of modern multi-tenant, multi-level data analysis. The research methodology was to survey of the status of security mechanisms in BDP systems and identify areas that require further improvement. Access control (AC) security services were identified as priority area, specifically Attribute Based Access Control (ABAC). The exemplar BDP system analyzed is the Apache Hadoop ecosystem. We created data generation software, analysis programs, and posted the detailed the experiment configuration on GitHub. Overall, our research indicates that before a BDP system, such as Hadoop, can be used in operational environment significant security configurations are required. We believe that the tools are available to achieve a secure system, with ABAC, using Apache Ranger and Apache Atlas. However, these systems are immature and require verification by an independent third party. We identified the following specific actions for overall improvement: consistent provisioning of security services through a data analyst workstation, a common backplane of security services, and a management console. These areas are partially satisfied in the current Hadoop ecosystem, continued AC improvements through the open source community, and rigorous independent testing should further address remaining security challenges. Robust security will enable further use of distributed, cluster BDP, such as Apache Hadoop and Hadoop-like systems, to meet future government and business requirements.

ACKNOWLEDGMENTS

I would like to thank my supervisors, Professor Changchun Zou and Professor Jun Wang for their wisdom, patience, and calm guidance as I progressed through the UCF PhD program.

I would also like to recognize all the support and encouragement I received from my family. They inspire me each and every day.

I am also very appreciative of the sponsorship of my employer, The MITRE Corporation, in earning this degree.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE: INTRODUCTION

We are in the era of Big Data where the volume of digitally generated, processed, and exchanged data is increasing at exponential rates. At the same time, attacks on computers and networks have become a critical issue in public and private business sectors. Research is needed where these two domains critical to modern computing intersect, i.e., Big Data Processing (BDP) and cybersecurity. Past research in this area has been fragmented, in that it is difficult to build out BDP experiments that comprehensively evaluate security. New focused attention in this area is needed since security of open source BDP was considered as an after-thought in its development. Reports of large-scale data breaches indicate the importance of addressing cybersecurity vulnerabilities before BDP systems can become operational.

This research focused on a survey of the cybersecurity of the BDP ecosystem, developed solutions to BDP cybersecurity research barriers, and demonstrated a framework for detailed analysis of BDP Attribute Based Access Control (ABAC). The Access Control (AC) security service was identified as a priority area for research because it is a fundamental underpinning for most other security services that are required to achieve robust layered confidentiality, integrity, and availability.

Thesis

The central thesis of this dissertation is as follows:

BDP security is at a nascent state given that traditional approaches to computer and network security were either not applied during initial design or fundamentally didn't fit with new distributed compute and data store methods. To advance research in this area a framework for experimentation is needed to enable BDP security investigations. We propose a data generation program that produces a synthetic data set that represents a multi-sensitivity level data that

could be accessed by users at multiple authorization levels. We demonstrate establishing this environment on a cloud service provider infrastructure with open-source software. This enables detailed analysis of ABAC and this framework can be extended for investigating other BDP security services.

Background

Many domains generate bid data sets and are applying distributed, cluster, parallel processing. For example, the management of computer and network system logs to analyze the occurrence of cybersecurity events requires big data processing [1]. Apache Hadoop[1] is considered the leading open source framework for distributed parallel cluster computing, i.e., BDP. As shown in Figure 1, the Apache Hadoop ecosystem consists of data flowing into a data ingestion, processing and storage environment to provide analytic insights to data visualization systems. Overseeing this process are security and performance management tools.

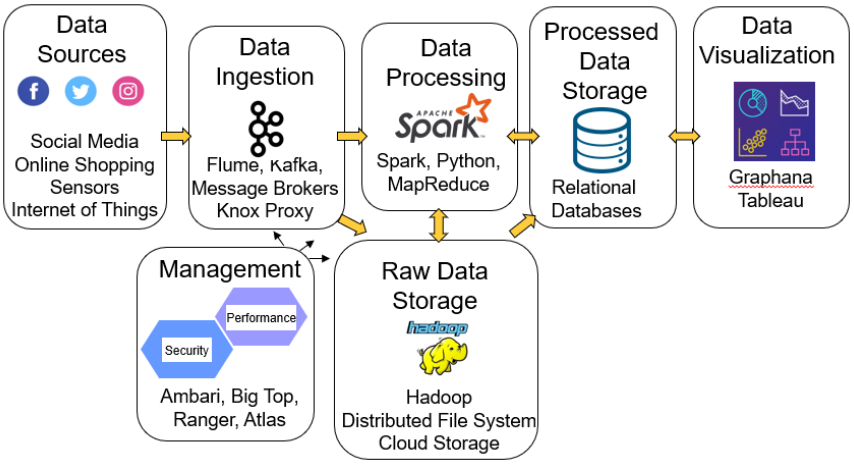


Figure 1: Overview of BDP Data Flow, Compute, and Storage Components

[1] https://hadoop.apache.org/

Data is ingested from multiple sources to message brokers, such as Apache Flume[2] and Apache Kafka[3] and through proxy gateways, such as Apache Knox[4]. The sources may include medical, transportation, industrial system sensors, social media data and a wide variety of other data types from Internet of Things (IoT) technologies. Message brokers allow processes to subscribe to the data streams of interest. In addition to storage in the Hadoop Distributed File System (HDFS), analysis is conducted using parallel data processing frameworks, such as Apache Spark[5] and MapReduce. Based upon popular programming languages such as Python and Java, these tools lower the difficulty bar for raw data analysis. After data is processed, Relational Database Management Systems (RDBMS) are used to execute Structure Query Language (SQL) analytics on data. Data visualization tools such as Graphana[6] and Tableau[7] are applied at the final stage to provide the graphical data-driven intelligence and business insights. This integrated collection of BDP systems can provide transformative information for businesses and can also present a broad cybersecurity attack surface.

The current transition is from monolithic single systems to distributed parallel processing environments with components developed and managed by a diverse open source and industry community. An integrated, consistent approach to security is challenging in this new framework. The large volumes of data that are coming into data processing and storage environments may contain data at various sensitivity levels. To control access to these large volumes of data, security services need to be extended to the raw data storage and parallel processing environment.

[2] https://flume.apache.org/
[3] https://kafka.apache.org/
[4] https://knox.apache.org/
[5] https://spark.apache.org/
[6] https://grafana.com/
[7] https://www.tableau.com/

Earlier RDBMS security methodologies, such as controlling access to rows, columns, or views does not map well to this unstructured data processing cycle. Rather than using schema-based permissions in BDP, AC policies need to be applied dynamically since big data systems implement the concept of a schema on job execution.

BDP security is the protection against unauthorized disclosure, modification or destruction using hardware and software techniques. An initial and key component of a security mechanism is the AC decision process. AC is a prerequisite to communication integrity, data at rest encryption, and other security services. It is distinct from the identification process in that AC enforces the policies, rules, and decisions on who has access to what.

For access control security services, Attribute Based Access Control (ABAC) is identified in many research papers as a leading approach. ABAC is implemented in a BDP environment by using central management of attributes and policies (using Apache Ranger[8] and Apache Atlas[9] for example) and enforcement of attribute-based policies at the cluster boundary (Apache Knox) and each ecosystem component. Attributes assigned to:

- Objects/resources, (data, such as healthcare and social media data)
- Users/subjects, (e.g., doctor, researcher, patient, social-media user)
- Provenance changes, (such as execution of programs that include data masking or sanitization).

ABAC provides greater flexibility to support dynamic security policies, than previous AC models, such as list-based AC.

[8] https://ranger.apache.org/
[9] https://atlas.apache.org/

Motivation

We observed during our research that many AC experiments do not include a large data set that represent the multi-tenant users, multi-sensitivity data use case. So, we have focused on this area to provide meaningful contributions.

Big data refers to situations where the volume of data is beyond what is traditionally stored on a single computer, (e.g., terabytes and larger), the variety of formats and structure does not lend to easy insertion into a schema (i.e., no-schema data), and the speed or veracity at which the data is generated, communicated, stored, and processed is high. In such big data systems, lines or files of data are appended to previously stored data rather than making modifications.

There is currently an emphasis on consolidating, centralizing, and interconnecting distributed systems to resolve hard problems. The goal is to reduce data inconsistency and enable application access to data. However, a much more dangerous security problem appears with this trend. As data silos become interconnected, unauthorized data leaks become more likely without well designed, integrated AC features.

Data processing can derive data at higher sensitivities. Linking, combing, and extracting data can enable derivation of more highly sensitive information.

During our research, an example use-case of a healthcare large data set was used to illustrate the implementation of AC. Healthcare data is subject to several national and international regulatory requirements. In the United States, laws include the Health Insurance Portability and Accountability Act (HIPAA) and Health Information Technology for Economic and Clinical Health Act (HITECH).[10] These laws and their supporting policies address the use and disclosure of individuals' health information by covered providers.

[10] https://www.hhs.gov/hipaa/for-professionals/index.html

Research Approach

Our approach to analyze BDP security has been to survey related research, review standards, and execute a prototype experiment using open source tools. We used the leading open source BDP approach, i.e., the Apache Hadoop ecosystem. Based upon our analysis, we identified AC, specifically ABAC, as the area for further detailed investigation. We made a data generation program, analysis programs, and detailed our experiment configuration. These details are posted on GitHub[11]. We developed recommendations to further advance BDP security based upon our Apache Hadoop ecosystem experiment. These results have been published in the referenced papers. Overall, we believe that continued improvements through the open source community should further address the identified security challenges. The organization of our research and progression from overall architecture analysis to implementation of an experiment that allowed us to discern specific findings that are documented in this dissertation is shown in Figure 2.

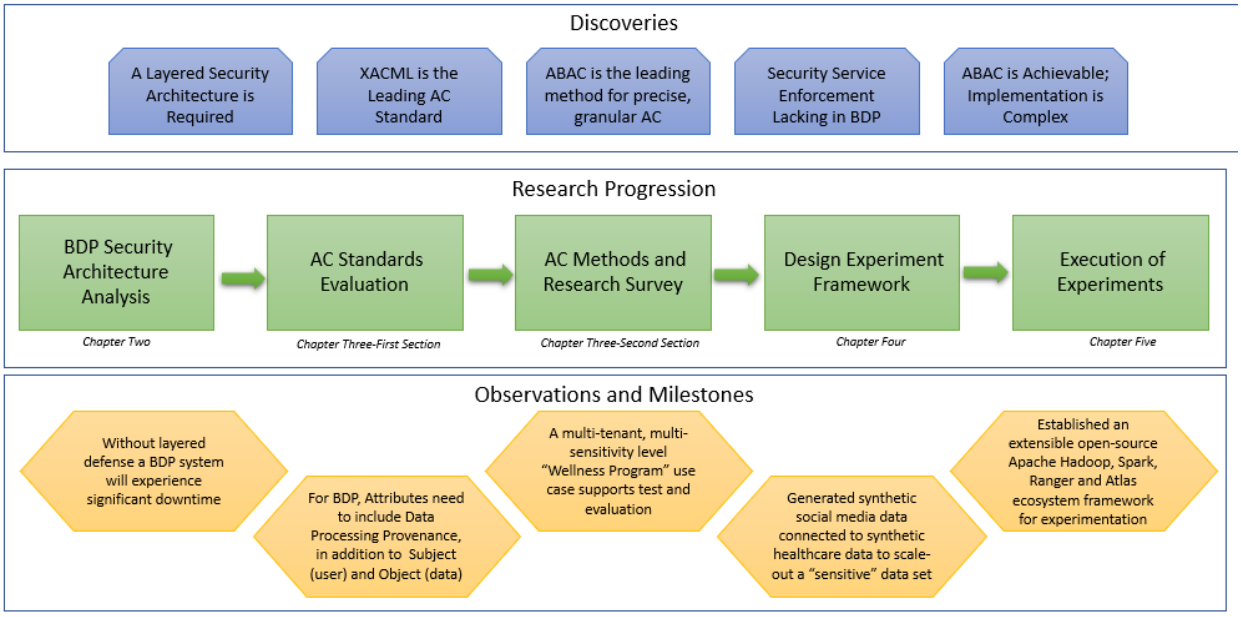


Figure 2: Overview of Research, Findings, and Organization

[11] https://github.com/AnneMT

Organization of this Dissertation

This thesis is organized into the following chapters:

- Chapter 1 (this chapter) provides the thesis of this research, an overview of the BDP environment, background information, motivation and overall approach
- Chapter 2 provides the analysis of the BDP security architecture and model that was developed to analyze BDP security services
- Chapter 3 includes the detailed survey and analysis conducted on AC and ABAC security services as applied in the BDP
- Chapter 4 describes the experimental prototype that was established to research BDP ABAC, including details on the synthetic data generator created to represent data at multiple sensitivity levels
- Chapter 5 includes the results of our executing our experimental prototype, mainly focusing on the security issues, potential cybersecurity attacks and approaches to mitigate vulnerabilities
- Chapter 6 lists and describes our recommendations and priority areas for improvement as the conclusion of this thesis.

CHAPTER TWO: BIG DATA PROCESSING SECURITY ARCHITECTURE

Statement of Prior Publication:

This work was previously published at the IEEE MILCOM 2021 conference, as listed in Reference 2.

Based upon our research, we recommend a layered security architecture for BDP and we developed a model to demonstrate the value of this approach [2].

Our recommendation BDP security architecture is based upon work in this area by standards group such as the National Institute of Standards and Technology (NIST) [3] [4], open-source projects, and industry initiatives [5].

Traditional security frameworks and architectures, such as Defense-in-Depth [6] , are still applicable, however, these principles are implemented in a new manner. A unique characteristic of BDP environments is that the analytics and tools introduced to derive meaningful insights are dynamic, uniquely developed for specialized purposes, and often open-source. In a locked-down sensitive data processing environment, this type of dynamic introduction of executable code is akin to leaving the system open to the malware. The traditional closed, controlled approach can result in a substantial data-lake investment that is accessible by very few data analysts.

The opportunities provided by open-source parallel processing BDP systems, such as Apache Hadoop, are exciting and complex due to the large number of components that comprise BDP ecosystem, [7] [8]. Maintaining the security of these systems requires not only an understanding of the core storage, compute, and resource management components, but also an array of components that provide additional services such as high availability, management of high data volumes flowing into and out of the cluster, scheduling jobs, providing security and others.

These applications provide different methods for accessing the same data. Therefore, it is critical that each component applies security in a consistent manner.

In a multi-tenant use case, stored data is shared across the organization (different mission/business groups and users) in a way that enables each organization to run their own applications (e.g., MapReduce programs, Pig jobs, Spark applications, HIVE, Hbase). Security services need to be configured so that each user is segregated from each other and able to access only their authorized data.

Architecture Analysis

Measuring the correct amount or level of cybersecurity that needs to be integrated into the architecture of large scale, diverse data processing systems has been a long-term challenge in the information security domain. Standards organizations have published guidance on cybersecurity measurement based upon best practice, consensus approaches, however many measurements remain subjective. Objective quantification given the dynamics associated with attacks and protection mechanisms continue to challenge computer, network system manager and administrators. The recently published IEEE Standard for Big Data Security [9] helps to improve the assessment of big data technology security protection mechanisms against business security. This standard defines a framework that consists of a portrait level and algorithm level approach. By standardizing business risk assessments, improvements can be made in sharing, evaluating, and predicting BDP risk posture and inheritance when interconnecting to other systems. However, the standard depends upon the assignment of risk based upon several subjective factors such as data sensitivity levels.

Other security measurement standards applied to the security architecture analysis included the Exploit Probability, Impact Factor, and Service Availability as defined by NIST [10] [11].

Quantifying these factors can be complex and selecting the correct scale based upon false precision can lead to inconclusive results [12]. Therefore, in this analysis we apply the guidance to use simple metrics that help to quantify observations of attack and security service effectiveness. Although complex interrelationships of attack paths and redundant detection, correction systems exist, these more complex situations were not incorporated into the model. Therefore, the overall results of the model were used for broad recommendations for a layered approach to security, rather than requirements for specific security service control or mechanisms.

As more quantitative data on attack and security service control countermeasures is made available from test or real-world events, more complex interrelationships could be included in the model, such as the cyber-attack analysis conducted by Liu, Xing and Zhou [13] using Continuous-Time Markov Chains (CTMC) to capture the interdependence of attacks. As more complex multi-step attacks are incorporated into a model, however the complexities associated with the details can limit the scope of the analysis which could limit the diversity of attacks considered and skew focus and acquisition towards a subset of the necessary security mechanisms. For example, Chen, Kalbarczy, Xu and Iyer [14] analyze vulnerabilities using a FSM approach and reached insightful conclusion on a few threat campaigns under analysis. Scaling out this of analysis at a high level of detailed fidelity while maintaining overall accuracy could be challenging.

Recommended BDP Security Architecture

BDP system security is different from other data processing systems, e.g., relational databases. BDP processing is characterized as an ecosystem, in that the various components, such as the Hadoop software library and the accessories and tools provided by various Apache Software

Foundation projects are independently developed capabilities, however they all work together to provide a complete data management and processing environment. This results in differences in the implementation, integration, and execution of security services in the following ways:

- Security services (mechanisms) need to be applied in a distributed manner in the data processing and compute, store layers, e.g., at the master node, each data node, and supporting ecosystem server (MapReduce, Spark, Hive).
- Security information, (policies and permissions), need to be managed centrally and distributed through trusted methods to all the components in the big data ecosystem from the management layer.
- Security decisions, such as identification, authentication, access control and system and communication integrity, are made at all ecosystem components, not only by boundary, proxy servers at the gateway boundary layer.

Motivated by these differences and based upon our survey of BDP security research, an architecture that employs security services at layers within the big data ecosystem is recommend [15] [16] [17] [18]. This recommendation differs from these previous efforts by approaching BDP security from a wholistic, layered, architecture approach, rather than addressing only certain components of a BDP ecosystem. This approach is depicted in Figure 2. The diagram summarizes the layers of the BDP architecture, and the placement security services in each of these layers.

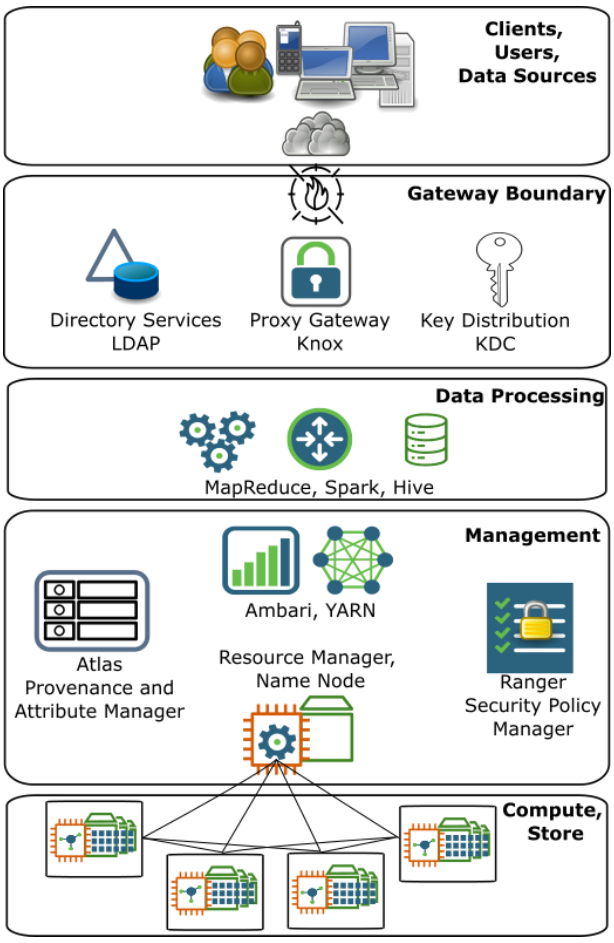


Figure 3: Recommended Layered BDP Security Architecture

Not all of the NIST RMF security service control families are implemented by the security mechanisms in the Hadoop ecosystem. For example, Awareness and Training (AT), Physical and Environmental Protection (PE) and Personnel Security (PS) security service controls are mostly external to a BDP system and satisfied through procedures and policies. A map of the NIST RMF [19] security control families to the BDP security mechanisms in the architecture are shown in Figure 3.

For example, Apache Ranger in the Management Layer and Directory Services (LDAP, AD) in the Gateway Layer provide the RMF Access Control (AC) services. Integrated together, a layered, defense-in-depth solution is achieved. The following sections further describe the

recommended BDP security mechanisms, and the security service controls they provide at each architecture layer.

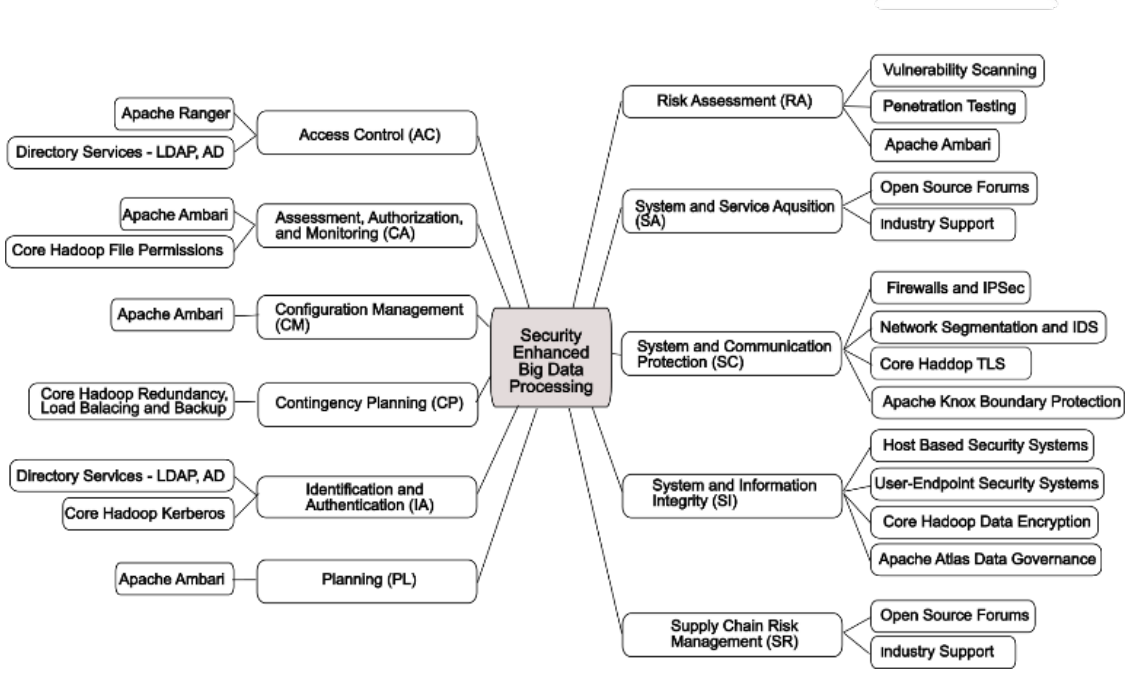


Figure 4: Mind Map: BDP Security Mechanism Mapped to RMF Security Service Controls

Gateway Boundary Layer

The BDP gateway, boundary layer builds upon traditional network boundary protection by providing application-specific gateway proxy services. Also, identification, authentication, and access control services through either a BDP-specific or enterprise-integrated directory service is a critical part of the boundary security services.

Typical products at this layer include Microsoft Active Directory (AD) or Open Lightweight Directory Access Protocol (LDAP) with a Kerberos Key Distribution Center (KDC) and Apache Knox application gateway proxy server.

Kerberos is the authentication mechanism integrated and optionally configured in core Hadoop. The original Kerberos implementation was developed at MIT and is currently available as open-source software. A feature of Kerberos is that because it is based on symmetric-key, keys used to

authenticate and encrypt connections are shared. Public Key Infrastructure (PKI)-based alternative approaches used in Transport Layer Security (TLS) use asymmetric keys managed by a Certificate Authority (CA) which overcomes the potential security challenge of shared keys; however, the processing level can be more intensive. Use of PKI to secure Hadoop has been investigated by researchers [20] however, we would expect that the most implementations are using the default Kerberos services.

At this architecture layer Apache Knox provides a stateless reverse proxy for a single point of access to the Hadoop cluster. It provides authentication, auditing, authorization for external users. It reduces the number of access points and can provide a single URL for accessing Hadoop services. This can provide security by concealing of Hadoop cluster installation details and data. Knox works with AD or LDAP server to authenticate users external to the perimeter [21].

Data Processing Layer

The data processing layer can consist of a wide variety of parallel processing and SQL to HDFS interfaces. Each of these can have their own Application Programming Interface (API) to authenticate and negotiate access to the distributed file system. This API authentication includes programs that stream data from external sources in to the BDP system.

Security at this layer depends upon configuration of access control, identification, authentication of permissions for users and their associated applications as well as the files (data and executables) permission settings. Researchers have proposed strategies to add security features to a library calls or modify to the data analysis programs (e.g., SQL on-Hadoop, MapReduce). A challenge with this strategy is that it needs to be coupled with strong controls that prevent the introduction of any unmodified or unauthorized analysis programs. Several important proposed concepts include query modification to extend access controls [22], rewriting queries to enforce

privacy aware access controls [23], and splitting execution of MapReduce programs between private and public clouds based upon data privacy policies [24].

Other techniques used to provide security at the data processing level include privacy preserving programs executing in parallel to mask sensitive data. Scaling out data anonymization techniques and tracking this as a sensitivity attribute enables enforcement of security policies so that sanitized data can be made available to users with lower authorization levels.

Management Layer

A robust management layer depends not only upon core Hadoop services, but also key ecosystem products to provide configuration, security policy, provenance and attribute management. Many of the security service control families are achieved at this layer through robust management tools, such as open-source Apache Ambari and commercially supported systems, such as the Cloudera Manager. Researchers have reported on the performance gains in an optimized, configured system [25].

In addition to management tools, the critical BDP management components include security policy management, such as with the Apache Ranger tool and data attribute life cycle, provenance management, such as with Apache Atlas.

Apache Ranger is the primary open-source framework for securing Hadoop. It manages the authorizations across the Hadoop ecosystem (HDFS files, Hive tables, etc.). Ranger uses Kerberos for authentication and TLS for encryption of data exchanged over the network. Highly granular, specific security policies can be defined and implemented across the ecosystem using Ranger [26].

Data provenance is defined as the record of the source, processing, and overall lineage of the data. These metadata attributes that track data provenance are critical to big data systems.

Traditionally data provenance is associated with audit logs and debugging. In Apache Atlas, data provenance can be expressed using a data model, business vocabulary, or other directed acyclic graphic terms. Making big data sets available for analytics in a secure manner requires tracking when process are executed that reduce the data sensitivity 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) [27].

Compute, Store Layer

The current Hadoop architecture uses metadata to handle the distribution and load balancing blocks across the data nodes. Like other file systems, the Hadoop File System (HDFS) uses a POSIX style Access Control (AC). The Apache Ranger project provides the hooks, using software plugin programs that are installed on each component, to manage access on each node, including Name, Resource, Job History and Data Nodes. Therefore, security AC checking is extended into the core Hadoop system, in a consistent, centrally managed manner. This achieves layered defense-in-depth. Several projects and commercial tools leverage the Ranger hooks to facilitate metadata management e.g., Apache Atlas, UC Berkeley Ground, and Cloudera Navigator. This provides the opportunity to integrate additional metadata into the AC decision. HDFS file system security is distributed across all the nodes in the Hadoop cluster. File permission settings are optional and by default disabled. When the file system permissions are disabled, anyone with access to the computer system node can do anything to the HDFS files. Also, encryption of files stored in HDFS is optional and disabled by default. Anyone with access to the local disk can read the unencrypted files. Data in the Hadoop cluster is exchanged in the clear, that is all network traffic is unencrypted by default.

The ability to disable file permissions, data encryption at rest and when exchange (transmitted over the network) between nodes highlights that security was added to Hadoop after initial development. Designing in security services at the beginning of the development process generally leads to an overall more secure design and reduces opportunities to bypass intentionally configured security services.

Basic security features are configured through settings in Hadoop XML files. Without these configurations, any HDFS account on any node in the Hadoop cluster is permitted access to their files anywhere in the cluster. Basic Hadoop operations where files are created in folders and Map-Reduce programs are executed with these files are open for any user to execute if the default security settings are not changed. The Hadoop fsck command allows users to know where blocks for any particular file are stored and can see the metadata to find all replicated copies of data. Cross system authentication is accepted, and users do not have to reauthenticate, e.g., provide a local system password, when reusing accounts across two systems in Hadoop.

Security Architecture Analysis

To analyze the proposed layered security service architecture, described in the previous section, a Finite State Machine (FSM) model of attacks and security service controls was developed[12]. This model demonstrates the use of BDP systems security service controls to thwart the impact of cybersecurity attacks and increase uptime. Overall, it provides insights on the value of security service controls.

A linear increase in security mechanism investment and maintenance results in a significant increase in uptime. The FSM used as the basis for the model is shown in Figure 4. It consists of 5 states and 10 transitions. In each state, the evaluation of different aspects associated with

[12] https://github.com/AnneMT

cybersecurity attacks and defensive security service controls was incorporated. Using best practices and reports from industry and academia, the likelihood of the attacks and defenses was considered. Specifically, random values, based upon the Binomial or Poisson distributions, were computed to represent the chance of attacks and likelihood the defenses were successful. The conditions and probabilities evaluated at each state are further described below.

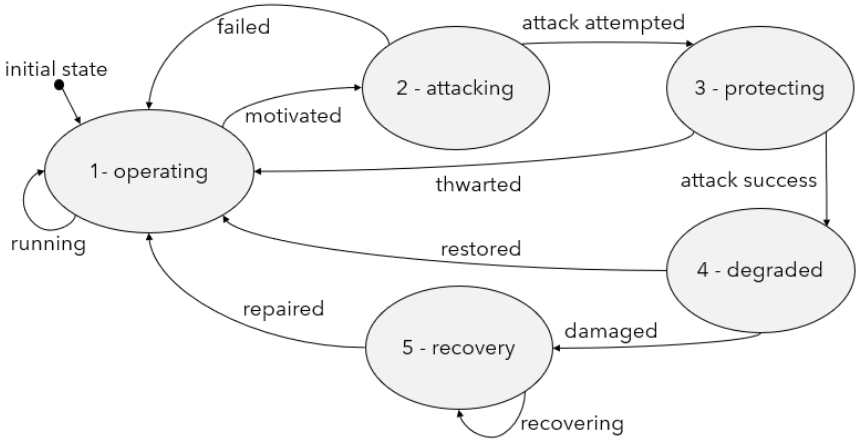


Figure 5: BDP Cybersecurity Attacks and Protections Finite State Machine (FSM) Model Operating

The initial state represents the BDP system in an active, operational state. The motivation of cybersecurity attackers are evaluated at this state. In this model, a transition out of the operating state was based upon two factors: (1) the value of the BDP system and data and (2) environmental conditions.

For the motivation based upon the value of the BDP system and the data it processes, we applied the Federal Information Processing Standards Publication 199 (FIPS PUB 199) standard System Categorizations (SC) [28]. The confidentiality, integrity, and availability (CIA) required of the system determine the SC, in accordance with formula (1):

SC information type = {(confidentiality impact),(integrity impact), (availability impact)} (1)

The impact of loss of CIA is rated as: Low = limited effect, Moderate = serious effect, or High = severe or catastrophic.

For example, a system with classified military system performance data would have a system categorization of high in all three CIA areas (SC = high, high, high).

The other factor, environmental conditions, have been identified in industry and academic research reports as motivations or triggers for attackers. Environmental conditions that we identified during this research were:

- Domain Name Service (DNS) name - systems with a specific DNS name are targeted by cybersecurity attackers, [29].
- Business Type - 90% of all attacks are about financial gain and espionage, so certain industries, are more at risk, [30].
- Political Climate - the internal and external political climate surrounding the organization that owns the BDP, such as recent layoffs could increase the likelihood of an insider attack, [31].
- Media Attention - media attention on the BDP owner, increases the likelihood of an attack, for example the COVID-19 pandemic has resulted in an increase in World Health Organization (WHO) related attacks, [32].

These areas were combined with the SC for an overall attacker motivation probability in the model. There are other factors that could be considered in evaluating the motivation of a cybersecurity attacker, however overall, most systems connected to the Internet today have experienced at least one or more cybersecurity attack, so we view an overall motivation factor of 80% or higher as reasonable [33] and used for transition from state 1 to 2 (1-2) motivated.

Attacking

This state represents the likelihood that a cyber security attack will be launched against, sent to, or executed on a BDP system. This state considers the likelihood and impact probability of approximately 350 different attack methods defined in the ATT&CK™ framework.

The DoD Cybersecurity Table Top (CTT) Guidebook provides the definition of cybersecurity attack likelihood and impact that we used as the basis for assigning probabilities to the attacks [34]. A scale of 1 to 5 was applied, with technically complex, low likelihood attacks were assigned probabilities in the 1 to 2 range and technically easy, well know and more highly likely attacks were assigned probabilities in the 4 to 5 range. A value of 3 was used for moderate complexity and likelihood. At this state, each attack was considered independent of the presence of security service controls or mechanisms that might thwart or otherwise neutralize and stop the attack.

The potential impact of the attack on the operation of the BDP system was also assigned on a scale from 1 to 5. A low value assigned for impact would indicate a cybersecurity attack would have little impact on the mission, for example, whereas a high value of impact could indicate a mission abort if the system were under cybersecurity attack. This value was used as the probability (p) in generating a Binomial distributed random value. If the resulting variate is "success" then the attack is considered "attempted" and there is a transition from state 2 to 3 (2-3) attack attempted to the protecting state. If the computed variate is "failed" then there is a transition back to the initial operating state.

Protecting

For each attack represented as successful, we evaluated the corresponding security service control. The mapping between NIST RMF security service controls and cybersecurity attacks defined in ATT&CK™ supported the assessment of protection measures [35] [36].

This evaluation considered the likelihood a mechanism to provide the security service control was implemented and maintained. For a BDP system such as Hadoop, the likelihood of a particular mechanism would depend upon the maturity, support, and investment in securing the system to an operational business grade status. Five cost model configurations of a BDP system, based upon a Hadoop ecosystem, were defined and used in the model as summarized below:

- Cost Model A – Default Hadoop installation
- Cost Model B – Use of core Hadoop security services and Operating System (OS) security
- Cost Model C – Enhanced with open-source security systems
- Cost Model D – Industry supported security systems
- Cost Model E – Enhanced (e.g., secure cloud) industry managed services

The resiliency and completeness of the systems security increase from A to E, with E representing a complete, layered security architecture with managed security services.

Of the seven levels defined in the Common Criteria (CC) Evaluated Assurance Levels (EAL) five were used in the model as the basis for assessing the strength of the service and assigning probability values for likelihood of implementation [37]. The CC EAL, as applied to the BDP system, range from EAL1: Functionally Tested to EAL 5: Semi-formally Designed and Tested.

The other consideration incorporated in the model for the probability of successful protection is the maintenance of the security service control. The DoD Cybersecurity Maturity Model

Certification (CMMC) [38] and the related Carnegie Mellon Software Engineering Institute Capability Maturity Model (CMM) [39] were used to guide the assignment of probabilities to maintaining the security mechanisms.

Given the complexities of setting up an open-source Hadoop ecosystem with many options for injecting, transforming, processing and displaying big data, the balance between security and flexibility can easily be focused away from system and data protection.

Approaches as characterized by Cost Model A would represent less mature processes and practices that include design decisions that sacrifice security over flexibility. Whereas configurations represented by Cost Model D or E would exemplify more mature cybersecurity processes and procedures.

The average of the probability of implementation and maintenance was used as the probability input to compute a Binomial distributed variate. If the "success" of the control in preventing the attack is computed the state 3 to 1 (3-1) thwarted transition is executed and the state is transitioned to operating. If the security service control random value is computed as "unsuccessful," the attack is considered successful and there is a transition (3-4) to the degraded state.

Degraded

In the degraded state, the BDP system is considered compromised by the attack. The ability to recover or be resilient without incurring down time is evaluated in this state. Effectiveness of mitigations is evaluated. This represents impacts, such as performance degradations or defacements that could be considered as damaging the reputation of the BDP owner. The two conditions evaluated in the degraded state are: ability to operate degraded and the impact of the degradation.

The ability to operate in a degraded state is a probability, that is the probability the BDP system can continue to operate in a configuration where the system has been subject to a cybersecurity attack, for example data is changed in an unauthorized manner, however, processes continue to execute in such that manipulated results can be detected and corrected. The ability to operate in a degraded state is computed from the average of three probabilities: the sophistication and impact of the attack and implementation of security service controls. Less technically sophisticated and low impact attacks are countered when mature security controls are implemented thus increasing the probability the system can operate in a degraded state. The probability of operating in a degraded state is represented by three conditions low attack probabilities, low attack impacts, and security control implementation. The probability calculated is input to a Binomial distributed variate generator to determine the transition from the degraded state (4) to either the operating state (1) ("success") (4-1) or the recovery state (5) ("failure") (4-5). The formulas (2 and 4) used to determine if the system can operate degraded are:

$$\textit{Operate in a Degraded State Probability} = ((1 - \textit{attack probability}) + (1 - \textit{attack impact probability}) + \textit{control implementation probability}) / 3 \quad (2)$$

$$\textit{Operation in a Degraded State} = \textit{True, when the Binomial distributed random variable generated from the Operate in a Degraded State Probability is True} \quad (3)$$

The impact of the degradation is calculated based upon the degradation value assigned to each successful attack. The sum of the degradation value ranges from 0 to 20, based upon the maximum of the individual values (1 to 10) and an amplification value based upon the volume of attacks [40]. Lower values are associated with attacks that result in minimal noticeable performance impacts and detected data damage or breach. Higher values are associated with more extensive performance slowdowns, data breaches and significant reputation damage, such

as external web site defacements. The resulting degradation value is computed for each Cost Model A-E. The formulas (4) and (5) used to compute the degradation value in the model are:

Degradation Value = Maximum (degradation value assigned to each successful attack) + Amplification Value (4)

*Amplification Value = Average Degradation Value for all the successful attacks * Volume Score, where the volume score ranges from 0 to 1 based upon the number of successful attacks* (5)

Recovery

Like calculating the impact of the degradation, each NIST RMF security service control mapped to an attack has an average recovery time assigned. The amount of time associated with recovering roughly corresponds to the complexity of the attack and maturity of the security service. The unit of time used in the model is hours, with values for each successful attack ranging from a minimal amount of time (one hour) to 48 hours (2 days). The average recovery time from the reference spreadsheet is used as input to a Poisson distributed random variate computation. The resulting recovery time is then added to the total down time summed for each Cost Model. After recovery is complete there is a transition (from state 5 to 1) back to the initial operating state. Formulas (6) and (7) used to compute down time in the model are:

Down Time = Maximum (Down Time assigned to each successful attack) + Amplification Value (6)

*Amplification Value = Average Down Time for all the successful attacks * Volume Score, where the Volume Score ranges from 0 to 4 based upon the number of successful attacks* (7)

Model Results

The result of running the model through 365 FSM cycles, 5 times, for each Cost Model A-E is shown in Table 1 and Figure 5. The results illustrate that a linear investment in security mechanisms results in significant improvement in reducing down time. The first two models (A and B) which only use the default and basic Hadoop security services are not resilient to cybersecurity attacks and down the entire time. The mid-cost model (C) is down a significant amount of time and also operates in a degraded state. The models with the most robust security configurations (D and E) experience the least amount of downtime. The most robust configuration, (E), also operates with the least amount of degradation. Since model (D) lacks the capability to fully stop attacks, i.e., only reduce attacks, it has a high degradation. Clearly, for mission and business critical systems, where down time or operating in a degraded state can have a significant impact on readiness and business continuity, there is strong justification for the most robust security architecture configuration.

Table 1: Degradation, Down Time, Up Time Averages

Model	Cost Model ($K)	Degradation (Total / No./ Avg.)	Down Time (days)	Up Time (days)
A	2,200	0 / 0 / 0	342	23
B	2,900	2 / 1 / 2	335	30
C	5,500	231 / 38 / 6	271	94
D	8,300	704 / 117 / 6	141	224
E	9,500	63 / 11 / 6	9	356

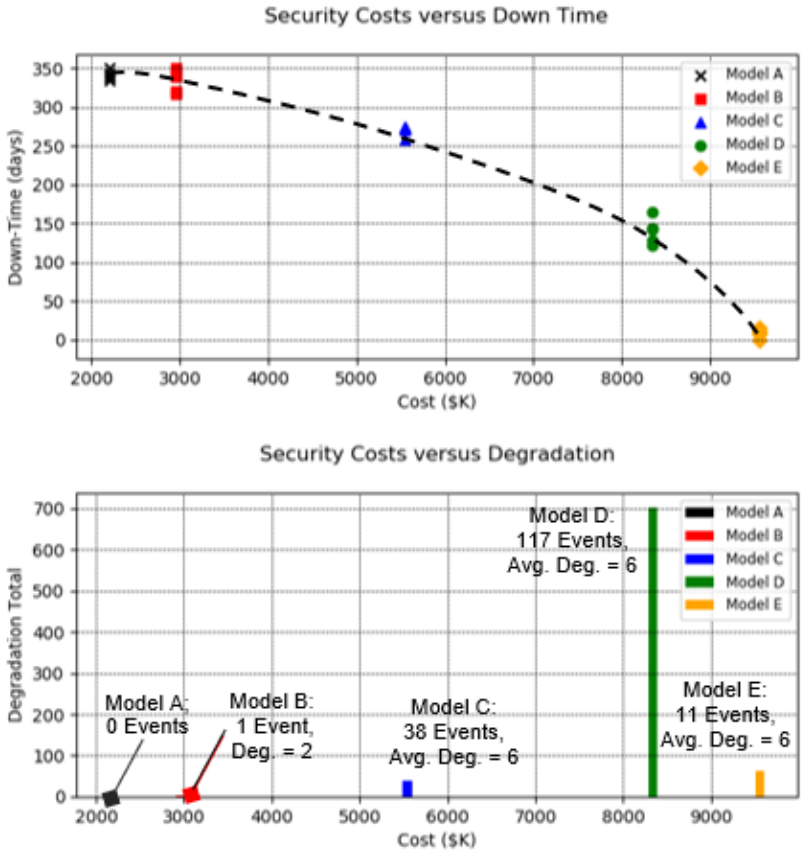


Figure 6: Model Results Indicate Linear Down Time Relationship to Cost and Significant Degradation Impact to Model D

Although the factors considered in this model are focused on the BDP system architecture, the concept of analyzing cybersecurity attacks and the resilience of the protection mechanisms to reduce down time could be applied to other information technology systems. This can help to support and justify investments in security mechanisms. Security system investment can be viewed as black hole, where an unlimited amount invested can appear to have negligible return on investment. However, this experiment demonstrated in an empirical manner that investment security mechanisms can have a significant increase in cybersecurity resiliency, i.e., resistance to cybersecurity attacks.

Architecture Recommendations Summary

In summary a layered security service control architecture is required for secure, multi-tenant BDP that includes:

- Enforcement of users and data sources authentication and access controls at the gateway (or proxy) boundary.
- Execution access control, identification, and authentication security services as well as control of process execution, such as incorporating privacy preserving programs as part of the data processing layer.
- Integrated security policy information and administration as part of the management layer, including data provenance and resource management.
- Policy decisions and enforcement on each compute-store node using local agent, client APIs synchronized with the security policy manager.

This approach was demonstrated by the model to have a significant improvement in reducing the effectiveness of cybersecurity attacks. In open source BDP systems, managers and administrators need to be proactive in configuring systems in a secure mode, since most default installations are unsecure. The model has shown that security must be implemented at many different locations, or layers in the architecture, from the user to the compute, store and management of the BDP system. If any of these layers of security are missing the system becomes vulnerabilities to many different attacks, as defined by ATT&CK. Without integrating either commercial or open-source security mechanisms into the BDP ecosystem the system is open to a wide variety of attacks to the point where it will most likely experience significant downtime.

CHAPTER THREE: ACCESS CONTROL SECURITY SERVICE ANALYSIS

As part of our BDP AC security services analysis, we conducted a detailed survey of standards and research in this area. The result of this analysis indicate that XACML is the leading standard and ABAC is the preferred approach. The following sections describe this analysis.

Access Control Security Service Standards

Statement of Prior Publication:

This work was previously published at the I/ITSEC 2019 conference, as listed in Reference 41.

Standards guide the implementation and future development of AC. As part of this research, we investigated the leading access control standards that apply to BDP [41]. Standard AC methodologies vary depending upon the level of granularity of information needed to make a decision.

AC Models

AC models were established early in computer system development as part of operating systems design and have evolved over time to address a wide range of use cases, including application-specific, multi-users and networked systems. An AC model provides a logical connection between AC rules and the mechanisms used to implement those rules [42] [43].

Typically, AC models are implemented in a layered manner within a computer-communications systems. The operating system and many application servers such as SharePoint typically provide AC services based upon a discretionary access control model (DAC) where users can grant access to others by configuring file permissions. However, in many domains, especially for specialized models and simulations, who-has-access-to-what is tightly controlled in a centralized

manner through the assignment of roles in a Role Based Access Control (RBAC) or Mandatory Access Control (MAC) model. This ensures, for example, that a database application enforces strictly defined, system enforced roles and responsibilities. To control access at a fine grain level, at the data schema or block level, a high-fidelity model is needed. The Attribute Based Access Control (ABAC) model uses metadata tags or attributes to achieve this level of control. The models are not mutually exclusive, in that various versions of the models are used in different components of the networked computer system environment. This concept of overlaying AC models and the associated implementing technology is highlighted in Figure 6.

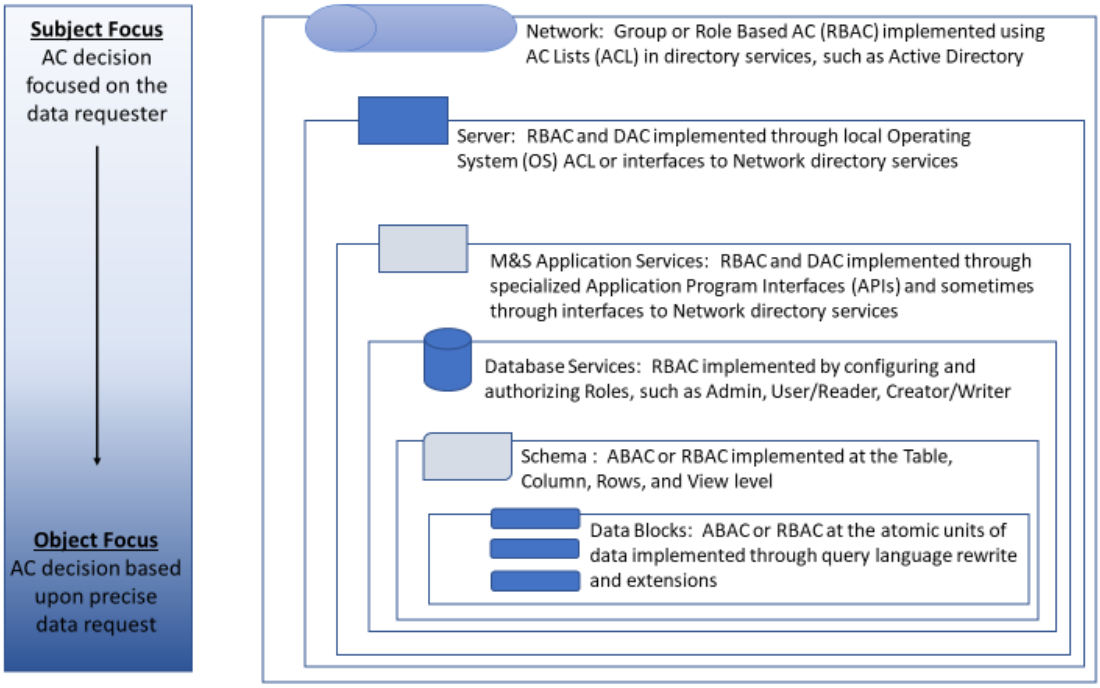


Figure 7: Layered Implementation of AC

At the network level, the focus is on ensuring a user is assigned to a group or role that is in an access control list. However, at the data block level, the focus is assigning permissions at an atomic data level. As AC policies and decisions are executed from the network to the data block

level. At the broad network level, the focus is on the user (device) requesting access to the environment. At the data block level, fine grain controls are needed, to authorize data producers and consumers access to specific data sets.

The widely used access control list (ACL) method focusses on listing all authorized subjects, assigning subjects to groups and granting group access to a list of objects. However, these lists are becoming rather large with the current explosion of data and access requests coming from not only human-users, but many processes and Non-Person Entities (NPEs) such as medical sensors and other Internet-of-Things (IoT) devices. To achieve a high-fidelity data block level AC, a different technique is needed. ABAC is a viewed as a method to move away from list-based AC information to enforcement of policy-rules based upon subject and object attributes or characteristics. Enforcement is based upon determining if the subject have the required attributes to access an object with certain attributes [44], [45]. With the next generation data control trend being based upon ABAC, the following sections focus on this AC model. The two primary standards for defining ABAC rules for AC are XACML and the relatively recently defined NGAC, summarized in Table 2. Both XACML and NGAC are focused on defining attribute-based AC control policy enforcement in a standard, interoperable manner.

Table 2: XACML and NGAC Applicability

Standards	Applicability
XACML - eXtensible Access Control Markup Language[13]	An XML-based specification language to express the security policies in terms of rules and the architecture for the access control process
NGAC - Next Generation Access Control[14]	A framework that defines AC in terms of data abstractions and functions based upon attributes associated with users, processes and objects

The standard AC architecture components are shown in Figure 7 below. Implementation with the retrieval of electronic heath record (EHR) data in Apache Hadoop HPC environment is highlighted. The sequences of steps that occur when a user requests access to data, e.g., EHR data, are highlighted. 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 gateway. Directory services such as Lightweight Directory services such as a 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.

[13] https://www.oasis-open.org/committees/xacml/
[14] https://webstore.ansi.org/standards/incits/incits4992018

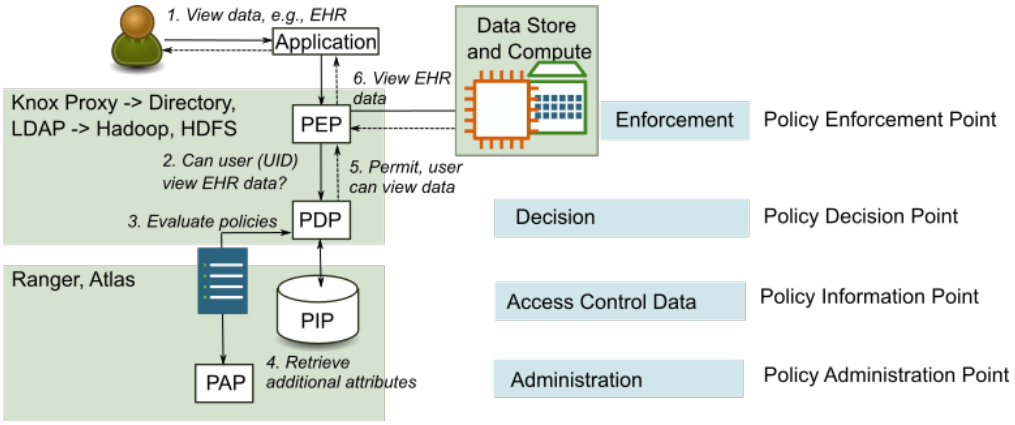


Figure 8: Standard Architecture Points that Contribute to the BDP AC Process

XACML

The Extensible Access Control Markup Language (XACML)[15] 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. It is the de facto standard for fine-grained ABAC. XACML defines three parts of an AC system: a policy language, request/response scheme, and an architecture. The policy language defines how to describe authorization constraints in an XML-based structure. The request/response scheme describes the protocol to send authorization requests and receive authorization permission decisions. The architecture contains three main components: enforcement, decisions, and management, as well as several supporting functions, information storage and retrieval. Specifically, the Policy Enforcement Point (PEP), the Policy Decision Point (PDP), Policy Information Point (PIP), Policy Retrieval Point (PRP), and Policy Administration Point (PAP). The core of the architecture is the PIP which loads policies (in XML-format) from the PRP and evaluates the authorization request intercepted by the PEP against those policies using additional information from the PIP when appropriate. The PDP passes the permission request response to

[15] http://docs.oasis-open.org/xacml/

the PEP which then permits/denies access to the requesting user/subject. The architecture describes decoupling the authorization decision into logical components that can be incorporated within the appropriate component, exposed at the necessary interface within the overall system architecture (e.g., presentation tier, web-application tier, data storage tier). This enables consistent enforcement across the multiple layers.

NGAC

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 [46].

NGAC is centered on configuration of relations. AC policies are enumerated based upon associative expressions. NGAC defines the expressions of the policy mode using four types of relationship configurations: Assignment, Associate (derive), Prohibit, and Obligations (dynamic). In NGAC- Generic Operations and Data Structures (GOADS) policies are expressed using the Z formal specification mathematical notation (ISO/IEC 13568:2002 - ZNOT). This is intended to enable validation and management of complex policies and relationships.

Both XACML and NGAC reference architectures specify four layers in the functional decomposition: enforcement, decision, access control data, and administration. Like XACML, the NGAC policy enforcement point 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.

Evaluation Of AC Standards

Fundamentally all AC decisions are based upon the making a grant or deny decision for a subject, (requesting user or process), to take an action on an object, (data or process resources). Complexity is introduced with the dynamic and non-traditional characteristics of the components in this decision. The requestor maybe a Non-Person Entity (NPE) and the decision to grant access to data maybe based upon what data that process has previously gathered (e.g., separation of duty issues). Environmental conditions such as time of day, location of the requestor, and legitimate relationships between data owners and requestors may all also be considerations. As a result, the AC mechanism for large, sensitive M&S data sets sensitivities need to have capabilities/features to handle these complexities.

Two key guiding principles that fundamental to the AC design are:

1. The AC model should be expressed in terms of a logical data model, e.g., a relational data model used in a RDBMS or relation attribute tuples used in large data sets
2. Name-based and content-view-based AC are both required, access decisions are based upon at least the subject and object.

Summarized in Table 3 below, and further detailed in this section, the two leading standards that implement RBAC and ABAC model policies, XACML and the more recently developed NGAC were evaluated against five key criteria. This is based upon the National Institute of Standards and Technology (NIST) guidelines and publications [4][5]. Although the NGAC standard provides potentially more robust technical services, especially with administration and management of AC policies both from the subject and object perspective, XACML has wider support and been adopted into more implementations. The technical benefits of shifting to

NGAC will need to be clearly appreciated to transition an installed base to a new construct. However, the data integration objectives and the data volumes are continuing to grow, so an approach better suited to the new era of big data management maybe timely for adoption, if the available supported products can be offered at enticing price points with acceptable transition strategies. In the primary areas of evaluated, NGAC provides advantages, however XACMLs widespread use may limit NGAC adoption.

Table 3: XACML and NGAC Comparison

Evaluation Criteria	XACML	NGAC
Security	Complexity makes deployment in a secure manner challenging, however increased use and experience may mitigate this risk	Ensuring complete secure deployment is technically challenging and with limited technical implementation references and community expertise
Policy Expression and Support	Supports some decentralized policy administration by an external delegation model	Objects/resources can be represented with minimal metadata, Weaker at handling environmental attributes and rules with a wide variety of attribute types, Supports history-based policies, and user-independent processes
Operational Efficiency	Less efficient, for each decision, policy loaded into memory and then evaluated	More efficient approach, where policy is loaded into memory at PDP initialization and updated as needed, enables linear scaling
Policy and Attribute Administration and Management	AC rule expression can become very complex, the standard does not define a methodology to review and verify permissions granted by subject or object and address delegations, overrides, and revocations. Metadata must be associated with every object/resource Strong at handling with a variety of attribute types within a trusted domain Does not address efficient policy review	Standard interface for attribute and policy administration Supports efficient algorithms for object and user review NGAC designed more efficiently in policy organization and execution Designed to handle more dynamic conditions By representing the process in the AC decision, NGAC provides greater policy flexibility and support NGAC provides more efficient policy review
Vendor Neutrality - Vendor Lock-In, Separation from Proprietary Operating Systems (OS)	In many implementations of XACML, the PEP is dependent upon the underlying OS	The NGAC definition enables near complete independence from the OS

Security

The most important area for consideration is the overall security afforded by the AC service. Security is achieved through reliable services that protect against threats to the AC services. The critical security requirements include:

- Safety property – ensuring that the execution of a sequence of manipulation operations does not result in access being granted out of compliance with the access control policy

- Data leakage / loss prevention – controlling the use of sensitive data within an organization to only those authorized and with a need-to-know by closely tracking/auditing sensitive data use
- Conflicts of interest management – identifying and preventing permission and access to data associated with organizations with competing or conflicting goals, activities, or objectives.
- Query privacy / Oblivious Transfer (OT) – avoiding the ability to infer information based upon the data access requests
- Bypass prevention - ensuring AC mechanisms, especially when implemented in client-side systems, cannot circumvent the implementation of the AC services.

Risk estimation to make security tradeoffs is an important consideration in the design. This is achieved in highly trustworthy approaches by using a Secure Context, information related to the execution of the data query is encapsulated to reduce risk. For example, for certain functions (e.g., select, insert, delete), argument elements are added (logically "AND"-ed using a WHERE function) to limit the returned data. However, query modification can have drawbacks that affect the correctness of the results and may also negatively impact scalability.

The same level of security should be enforced no matter how the data is accessed. This is achieved in some implementations submitting all service requests (e.g., read, search, write) through a gateway. In some implementations the SQL query statement is rewritten at the gateway to incorporate AC features. A single point of access can also enable single entry and synchronization of AC policies across distributed data stores logically located behind the gateway.

Policy Expression

Policy expression has to do with the scope and type of AC policy model supported. For example, the ability to support dynamic separation of duty. Flexibility in the expression and enforcement of permissions allows for: deny overrides, permit overrides, and first applicable based on order of authorization and/or policy processing. As described in the previous section, there are a wide variety of AC models that apply to different use cases and layers within a M&S computer, communication system environment, so as a result, not all technical approaches lend themselves to cover all conditions. However, ideally the selected approach allows for sufficient flexibility for expressing a wide range of models in both a centralized and decentralized manner.

Operational Efficiency – Performance Impact

The processing overhead for AC decisions can be significant. The algorithm selected for identifying and applying the applicable policy ideally supports linear (rather than exponentiation) scaling as the number subjects and objects increases. Organizing the policies in a manner that allows for loading in memory only the subset applicable to a decision request helps achieve performance requirements. Selecting the policies applicable to the target environment, includes addressing combinations from different authorities, overrides, and handling of conflicts. Policies and the subject/object/action attributes could be organized in various ways such as in a hierarchical graph format. However, the organization can directly impact the performance for loading into memory and processing the AC decision request.

The other factor in considering operational efficiency is if the system architecture model enables externalization of AC policy authorization decisions from within an application to a networked resource. By externalizing the AC decision, the AC service could be potentially used by multiple applications, thus potentially reducing management/maintenance overhead. However, the

responsiveness of the external service would need to be scaled to ensure performance requirements. The representation of requests from multiple applications needs to be consistent or standardized to ensure consistent execution.

Performance can be analyzed during three different phases of the AC decision process:

1. Loading AC policies into memory for processing
2. Finding the appropriate policy applicable to the access decision under evaluation
3. Computing or processing the policy to output an access decision

Performance in these three areas is directly driven by the ability to concisely and flexibly define policies so they can be succinctly segmented into manageable portions that can be efficiently processed. Verbose expressive languages to describe policies can be contrary to this goal. Concisely expressed notation can achieve more efficiencies.

Policy and Attribute Management

The complexity of maintaining and checking the integrity of policies and attributes can be a differentiator in the selected approach to AC. A user friendly, intuitive design can increase security by helping to avoid configuration errors. The treatment of attributes can become an unwieldy, confusing bottleneck in the AC decision process, creating information management challenges larger that the dataset/database the AC process is intended to protect.

The difficulty of the AC policy and attribute management approach is addressed through:

- Support for administrative review and integrity checking of AC policies and attributes assigned to subjects, objects and actions.
- Ability to discover resources by reviewing the granted access privileges for subjects and objects

- Fine grain administrative management and controls of who can (create/modify) administer policies, including the ability to delegate and inherit AC administration responsibilities across related targets (i.e., the policies; policy sets that apply to the subjects, objects, actions and environment within the elements of the AC schema)

Standards can leave the area of policy and attribute management as an implementation-specific decision. However, to reconcile access privileges across multiple authorization authorities/officials in a distributed environment, a means to harmonize policies and attributes needs to be conducted in a standard agreed upon manner.

Vendor Neutrality Versus Vendor Lock-in

Vendor neutrality versus vendor lock-in is directly related to the completeness of the applicable standards. Specifically, this issue has been associated with the Policy Enforcement Point (PEP). Standard interfaces to multiple PEPs from multiple applications provides greater flexibility. One-to-one interface from an application to a single PEP constrains a solution to a traditional operating system access control model.

Also, the administration of policies across a distributed environment with multiple authorities needs to be extensible in a standard way to avoid vendor lock-in. For example, workflow, calendar, and records management applications may need interfaces to several PEPs. Tight coupling to an operating system limits the ability to use an integrated AC service across multiple networked applications.

Policy Expression

In this section we compare the expression of an example medical data AC policy using the XACML and NGAC standards. The focus is medical healthcare data with defined security codes and a sample community of users with assigned attributes.

Representative healthcare data can be generated using Synthea™ [16] [47] synthetic patient medical data generator. Data is generated in the Health Level Seven International (HL7) Fast Healthcare Interoperability Resources (FHIR) specification format.[17] In this example policy scenario there are users in different roles with different levels of trustworthiness and need-to-know. Specifically, Doctors who have a relationship with a Patient have access to all the data associated with the user identifier (UUID). However, Researchers have access to the data only after it has been sanitized, that is meet an obligation for redaction of the certain data fields. HIPAA guidelines specify that de-identification technique mask 16 direct identifiers (e.g., names, email addresses, social security numbers, etc.) and that quasi-identifiers be generalized (e.g., remove specific date from a birth date and providing only the month and year). The overall challenge with this scenario is controlling access sensitive healthcare data while also making it available to researchers for M&S applications. The key requirements for the policy are:

- Make the healthcare data available to doctors in a legitimate relationship with the patient
- Make redacted healthcare data available to researchers, with the agreement the data will not be retained or reused.

[16] https://github.com/synthetichealth/synthea/wiki
[17] https://www.hl7.org/fhir/security.html#binding

Applying an Attribute-Based Access Control (ABAC) model in this example, a user requests to perform operations (read, write) on objects, EHR data. That user's access request is granted or denied based on a set of access control policies that are specified in terms of attributes and conditions. The attributes include security tags, environment conditions, and user and object characteristics. Attributes are input to the access control policies decision process that determines the operations a user may perform on a Resource (in FHIR) or object (in ABAC). In this example, we focus on using an attribute to specify that the identified data (object/resource) is not to be further disclosed without explicit consent from the patient. Core security labels defined in FHIR that could be used for example are:

- Context of Use, Purpose of Use: HRESCH (Health Care Research) and PATADMIN (Patient Administration)
- Data Sensitivity, Confidentiality Code: U (Unrestricted) and R (Restricted)
- Control of Flow: DELAU (Delete After Use) and NOREUSE (Do Not Re-Use)
- Value Set Obligation Policy: MASK, REDACT

As demonstrated in the Analytics on eXtremely Large European (AXLE) data project, [48], the data request response project includes processes that apply security labels based upon labeling rules, then make the access policy decision, and meet any required processing obligations before providing the result to the requester. This is illustrated in the Figure 8 below.

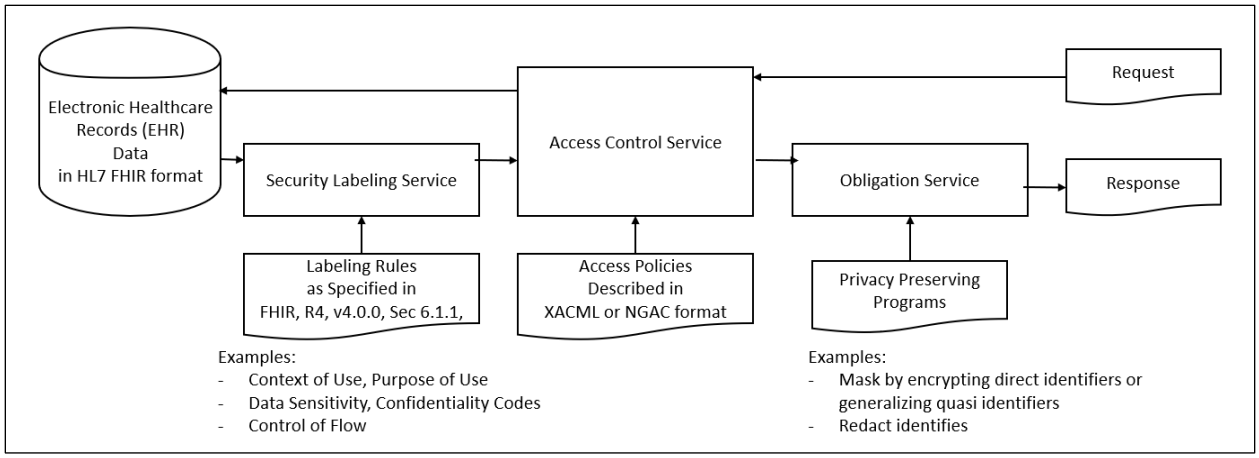


Figure 9: AC Policy Implementation Process with Example AC Service Request and Response

The ability to express the policy in XACML and NGAC formats was investigated using overarching ABAC guidance from NCCCoE – NIST [49] the OASIS XACML standard, the NGAC standard and several reference implementations. Example policies for medical data scenario are highlighted in the following sections.

XACML Example Policy Scenario Implementation

Several open source and commercial XACML reference implementations are available that support generation of XACML files from input policies. These provides a starting point options for application developers to incorporate ABAC features. For example, AuthzForce[18] is an open source implementation and Security Policy Tool[19] is a commercial implementation with a free trial version. The complete XACML file generated to fully implement the policy is too long to incorporate in this paper, however an example of the XACML rule based upon the matching attributes associated with the doctor is shown Figure 9 below:

[18] https://authzforce.ow2.org/
[19] https://securitypolicytool.com/

```
<Rule Effect="Permit" RuleId="rule_1">
  <Target>
   <AnyOf>
    <AllOf>
     <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:http://www.w3.org/2001/xmlschema#string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Doctor</AttributeValue>
      <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0::subjectcategory:accesssubject"
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:Role" DataType="http://www.w3.org/2001/XMLSchema#string"
MustBePresent="true"></AttributeDesignator>
     </Match>
     <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:http://www.w3.org/2001/xmlschema#string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">01-19d82286</AttributeValue>
      <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0::subjectcategory:accesssubject"
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:Legitimate Relationships"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"></AttributeDesignator>
     </Match>
     <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:http://www.w3.org/2001/xmlschema#string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Any Value</AttributeValue>
      <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0::attributecategory:resource"
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:Resource Type" DataType="http://www.w3.org/2001/XMLSchema#string"
MustBePresent="true"></AttributeDesignator>
     </Match>
     <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:http://www.w3.org/2001/xmlschema#string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">01-19d82286</AttributeValue>
      <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0::attributecategory:resource"
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:Single UUID" DataType="http://www.w3.org/2001/XMLSchema#string"
MustBePresent="true"></AttributeDesignator>
     </Match>
…. several additional match requirements were omitted for brevity …
     </Match>
    </AllOf>
   </AnyOf>
  </Target>
 </Rule>
```

Figure 10: XACML Policy Expression Example Subset Generated by the Security Policy Tool

The XACML standard includes standard method for tagging a wide variety of attribute values and matching, comparison and evaluation functions above. Although the XACML format is verbose, it is readable by developers and commonly used in a number of applications.

NGAC Example Policy Scenario Implementation

The guidance in the NGAC Functional Architecture (FA) specification uses a diagram to illustrate the policies and relationships. Rule generation is further specified in the NGAC Generic Operations and Data Structures (GOADS) using upon objection relationship notation as tuples. An open-source reference implementation for NGAC is the NIST Policy Machine - Harmonia Project,

however the code for this project appears to no longer be available.[20] An example figure based upon the medical data scenario is depicted in Figure 10 below. The graph illustrates the derived privileges for the example ABAC scenario, which are expressed in tuples as: (Wilton, r, FHIR Single Record), (Alice, r, FHIR Single Record), (Alice, w, FHIR Single record), (Bob, r, Conditions).

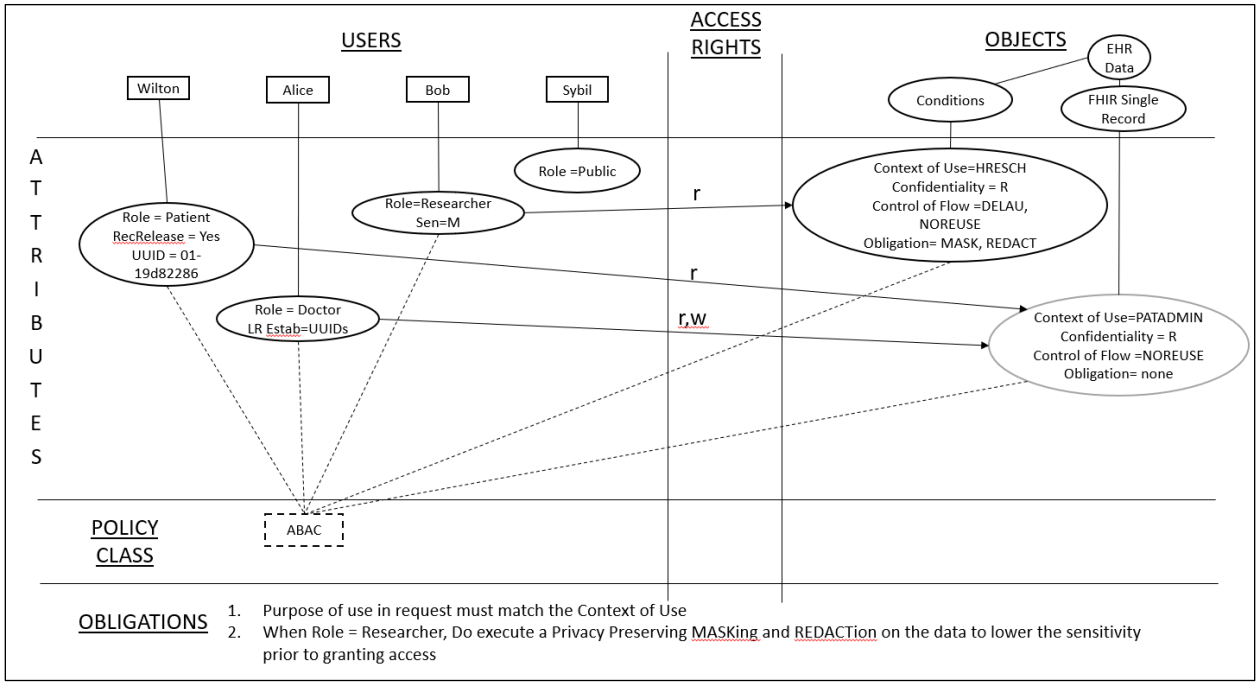


Figure 11: Example NGAC Assignment and Association Graph

AC Standards Summary

The protection of big data sets requires an access control solution that is extensible to distributed systems, managed by multiple authorities, and based upon mature standards. XACML is the currently the most widely used standard to implement ABAC, the leading approach for schema on search big data sets. NGAC provides many technical advantages to manage the complexities associated with a large, complex set of attributes for the large number of subjects and objects in a

[20] https://github.com/PM-Master/Harmonia-1.6

big data environment. An integrated AC approach at the network, system, and data storage level will most likely require applying several AC models as the AC focus shifts from controlling subjects (users) to controlling access to fine grain data objects. XACML and NGAC based reference implementations and products are available to address these challenges. A researched, thoughtful design that applies a standards-based approach helps developers to build upon progress in this domain and enable extending the security coverage as the data sets continue to grow.

Access Control Security Service Research

Statement of Submission for Publication:

This work will be submitted for publication in the Wiley Journal: Security and Privacy.

Access control (AC) services used in public cloud and private high-performance cluster-computing (HPC) environments are evolving to handle big data processing (BDP) and address privacy concerns. Attribute based access control (ABAC) is a consensus approach for fine-grain AC where large user-groups are permitted access to big data sets in a manner that meets stringent security requirements. The ability to provide access to more users with lower privileges or levels of trust requires data anonymization and other privacy-preserving techniques. This is a shift from traditional user-group and file system role or list-based AC methods. We analyzed and summarized the approaches and proposed the idea that robust AC for BDP needs to be based upon the attributes associated with the user (subject) and data (object), as well as the data provenance, i.e., a lineage attribute.

The overall approach for ABAC is defined in the National Institute of Standards Special Publication (NIST SP) 800-162 as the ABAC trust chain [50]. For the BDP ecosystem, attributes such as data processing lineage and environmental conditions should be added as part of the AC trust chain, as depicted in Figure 11. 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.

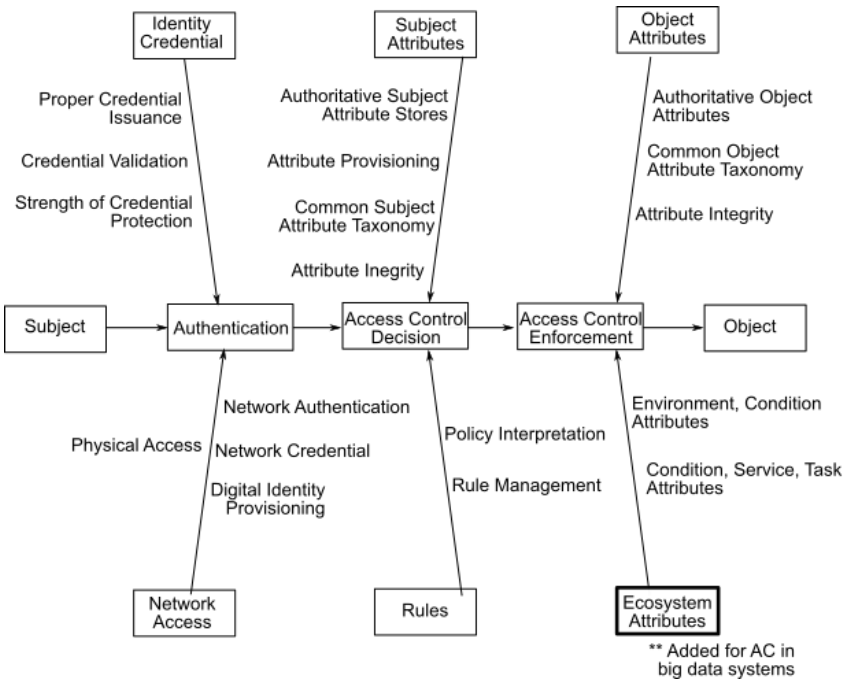


Figure 12: Modified NIST SP 800-162 ABAC Trust Chain

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 upon identity credentials and network access permissions. The AC decision is then initiated based upon 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 include the environment attributes. In this step, data (objects) 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 these 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.

The architecture of the components used to implement the AC trust chain in BDP needs to be layered to provide defense against various threats. As with other security services, AC requires a defense-in-depth approach. Several researchers have proposed the application of AC at various points in big data ecosystems. Most are based upon the standard approaches discussed in the previous section, especially XACML.

Overall, there is a lack of consensus on ABAC details, which make 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 large a 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 avoid bypasses, and a consistent and synchronized implementation are challenging areas explored by researchers. There are several published research articles and open-source projects on various aspects of BDP AC using ABAC, thereby contributing to a consensus of approaches to ABAC implementation.

Approach

In this section, we describe three technical areas that 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 a 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 sections, research on data provenance is reviewed and techniques to track the execution of privacy preserving programs and lower the data sensitivity are 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 is 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 the healthcare outcomes. Figure 12 illustrates the data exchanges between roles in this use case.

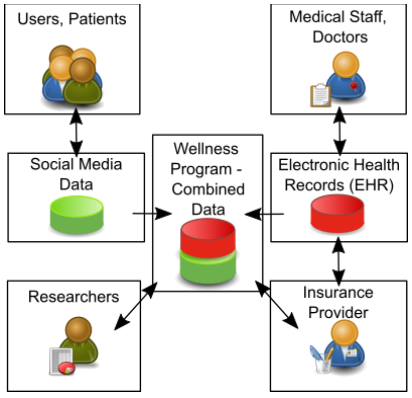


Figure 13: A Multi-Tenant, Multi-Level "Wellness Program" Use-Case

AC Policies

A 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)[21] and industry regulations, such as the California Consumer Privacy Act (CCPA)[22] 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., described the development of LEGALEASE and GROK for building and operating a system to automate governance policy definitions [51]. 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

[21] https://www.hhs.gov/hipaa
[22] https://oag.ca.gov/privacy/ccpa

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. As a challenge of this project, 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 [52]. This covers the software development lifecycle, including the secure coding practices of the Open Web Application Security Process (OWASP)[23]. In support of AC, the data processing lifecycle is managed using labels and verification checks.

The predominant vendor-neutral standard for AC rule definition, XACML, defines a policy language that specifies how to describe authorization constraints in an XML-based structure. NGAC is an important evolution in AC standards for consistent implementation of complex static and dynamic AC policy definitions in data-intensive systems. NGAC is specifically designed to more completely address areas such as contextual (e.g., environmental) factors and obligations at the policy enforcement point, in addition to dynamic additions, removal, and relationships between subjects and objects [53].

[23] https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/migrated_content

Considering the healthcare use case, 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. For example, Table 4 illustrates policies that require AC model flexibility in defining the subject, object, and environment policies and attributes.

Table 4. Example Big Data AC Flexible Policy Statements

Access granted based on subject role relationships: • Legitimate relationships among doctors, medical staff, and patients, under the care of (patient of) a provider • Hierarchy of roles, medical staff can inherit their access permissions or doctor-designated specialists for certain conditions
Access granted only during certain times: • During a doctor's appointment • At a specified frequency, e.g., weekly, monthly • During an emergency (override DENY level situation)
Access granted under certain conditions or context: • Executing programs that generate only limited data to control the volume of data that are accessed • Explicit consent, "Opt-In," by the patient, system user, or owner
Access for research only after sanitization or anonymization programs (obligations) are first satisfied

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 upon the use case scenario influencing the authoritative source for the attribute, metadata values, and their dynamics during the data lifecycle. NIST provides guidance and considerations for defining attributes that are reliable, well-informed, and maintainable [54].

There are diverse ways to express the attributes used as the basis for an AC policy enforcement. Attribute specifications and management are heavily influenced by the use-case and domain applied. However, there are overlaps and similarities in attribute schemas that provide the opportunity to identify optimizations. Several attribute definition strategies have been demonstrated in Hadoop processing environments, some of which are at the conceptual stage. Recently published research on approaches to attributed-based AC policy enforcement are summarized in Table 5.

Table 5: Analysis of ABAC Approaches in BDP

AC Method (and References)	Description	Security Observations	Performance Impact
Provenance, History Based [55] [56] [57]	The provenance of the data (meta-data) is used as part of the access control decision process.	Supports ensuring execution of privacy preserving programs	Overhead from updating data attributes should be manageable
Resource Based [58] [59]	Permissions are assigned directly to the data. For hierarchically organized data, permissions are inherited by sub-folders from superior folders. The data permissions management is integrated with the user/role ACL permissions. Expands upon current file system permissions through centralized management and increased controls on inheritance.	Strengthens previous methods by adding controls on data processing resources	Enhanced meta data management strategy minimizes the overhead from permission checks
Task - Role Based [60] [61] [62]	Tasks (read, write, execute) are associated with certain roles. Security policies are enforced based the role a user is assigned and the tasks that role is authorized to perform. An example is in healthcare where a patient conducts certain medical tests at home and in which data are written into their medical record. The patient is authorized only to execute a certain task and write data. Applied to IoT.	Strengthens current methods by adding controls to tasks	Central AC management minimizes performance impacts and overhead on subjects (clients)
Relationship Based [63] [64] [65] [66]	The prominent method used in online social networks, where authorization policies are based upon relationships. Users grant access to information based upon relationships, such as "friends." The focus of current published research is on mining data to derive the social relationships.	Distributed controls to a broad set of users leads to inconsistencies	AC method has limited impact on speed of data retrieval
Role Based, Time Bound [67] [68]	Enables temporary, time limited privileges to be provided to users assigned to roles. Enables a just-in-time access model. An example is limiting Wi-Fi access in a public area. For big data systems, access to certain users, such as researchers, could be limited to business hours.	Risk of unauthorized access constrained by time	Limited additional overhead
Object Tagged, Rule Based [69]	Meta-data tags used as part of the access control decision process.	Ensuring integrity of attributes critical to trustworthiness	Overhead from adjusting attributes appears low
Semantic and Ontology Role Based [70]	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	Overhead from analyzing and matching attributes could impact performance
Content-Sensitivity Based [71] [72]	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. This is a version of the semantic-ontology method.	Technically complex, depends upon accurate inference	Overhead from analyzing sensitivity score could impact performance

Identity standards, such as the Security Assertion Markup Language (SAML)[24] developed by the Security Services Technical Committee of OASIS and implemented in Shibboleth[25], provides standard attribute definitions [73]. 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[26], and the National Identity Exchange Federation (NIEF)[27].

Using the healthcare use case, as an example, SAML v2.0 for healthcare defined attributes [74] [75] are input to the AC policy decision process. This is illustrated for a big data, Hadoop ecosystem in Figure 13 below.

[24] http://saml.xml.org/
[25] https://wiki.shibboleth.net/confluence/display/ CONCEPT/ AttributeNaming
[26] http://docs.oasis-open.org/security/saml-subject-id-attr/v1.0/csprd02/saml-subject-id-attr-v1.0-csprd02.html#eduPerson
[27] https://nief.org/attribute-registry/

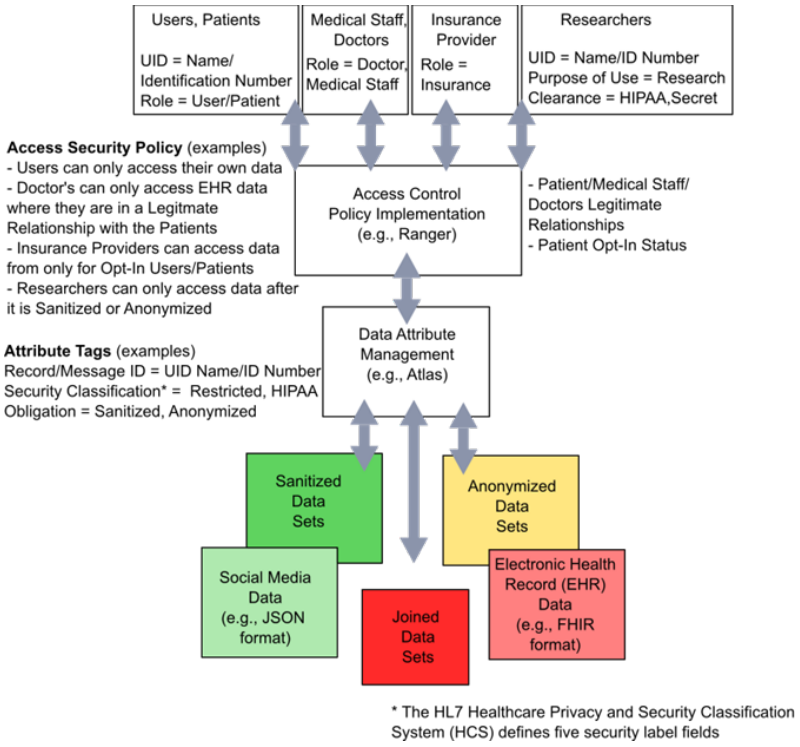


Figure 14: Attributes Assigned and Managed to Support Healthcare Use Case AC Decisions

Encryption provides a 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 [76] [77] [78].

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) [79].

Hellerstein, et al. [80] introduced 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 [81] [82] [83].

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 [84].

In the next section, we discuss the experiment we conducted by updating the data security classification and obligation attributes over the course of a represented data processing lifecycle to support provenance tracking.

Analysis Strategy

We propose evaluating AC methods based upon two primary areas: security and performance. Historically, there is considerable debate on the approach used to measure the security and performance, and thus this analysis serves as a framework of considerations for application to a use case.

Security

The strategy for evaluating system security involves two areas: system security threats and countermeasures strength. AC policy verification and test tools have been proposed and developed to help manage this complexity [85].

Techniques used to attack AC systems are identified using the Adversarial Tactics, Techniques and Common Knowledge (ATT&CK™) knowledge base, as listed in Table 6. These tactics are based upon actual computer and network attacks documented by 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 [86] [87] [88].

Table 6: 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)[28] are identified on the Apache Hadoop site. The total number on the CVE[29] 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 location of sensitive information in temporary files, privilege escalation, security values handling issues, and command injection. Most corrections to address the CVEs requires 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. In the area of access control and authorizations,

Although there are several security standards to guide 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[30] 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.

Performance

Evaluating the performance of AC services, the big data processing environment involves altering the placement and configuration of the AC services under the same job execution.

[28] https://hadoop.apache.org/cve_list.html
[29] https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=hadoop
[30] https://issues.apache.org/jira/browse/RANGER-1973

TeraSort[31] is a commonly used Hadoop benchmark, for example. It consists of components to generate random data, sort the data using a MapReduce program operating on a Hadoop Distrusted File System (HDFS) cluster, and validate the output. The performance of each ecosystem component could be measured in terms of elapsed time for process execution, processor utilization and bandwidth; however, the overall goal is to decrease the time between job submission and response, measured with consistent hardware configurations, network bandwidth, data size, and overall data processing objectives.

AC service components that may vary varied during benchmark runs conducted to determine the performance are summarized in Table 7.

Table 7. Summary Performance Analysis Areas

Approach	Area to Adjust to Analyze Performance
Knox Proxy server, for authentication and user access control	Number of users and number of processes being submitted to the proxy servers
Ranger Policy Manager and Directory Server	Types of users (subjects), such as a large number with various privilege limitations, number and type of subject attributes
Atlas Data Provenance Server	Various data sensitivity types or levels, as configured as object attributes
Privacy Persevering Algorithms	Varying the type and placement and execution of PPDM in the data processing lifecycle.

Quantifying the differences is supported by tracking the elapsed time for a process execution, processor (CPU) performance, and memory utilization, which are all reported through BDP management interfaces, such as the Hadoop Web user interface or Apache Ambari. Although others have analyzed the performance of alternative ABAC configurations independent of their impact on the BDP, we believe integrating a realistic multi-tenant, multi-level configuration on a

[31] https://hadoop.apache.org/docs/r3.0.0/api/org/apache/hadoop/examples/terasort/package-summary.html

BDP system such as Apache Hadoop will provide additional baseline performance measures [89].

In the BDP, maintaining the data attributes, (metadata), as the data are processed in a trusted, distributed manner is a complex task. Decentralized distributed policy enforcement can make revocations and updates extremely challenging, particularly if there is any policy synchronization latency in distributed systems subject to different management priorities. The status of the data processing lifecycle workflow, process service command execution, and environmental factors such computing and communication status can all result in contention for resources in updating AC policies.

Information about strategies used to measure the performance are available from published sources listed in the references and from blog sources such as [90]. The objective of the performance analysis is to verify if the AC services create a processing contention based upon allocated virtual cores, memory capacity issues, high data reads and writes, and/or high network bandwidth use. Impacts from complex AC policies, including user role attributes and metadata proliferation can also be considered in a performance analysis. Additional insights would be gained by including a large dataset that examines capacity challenges, including replication and block movements in the face of Data Node commissioning and decommissioning. Performance statistics that can be obtained from the Hadoop Web GUI are listed in Table 8.

Table 8. Hadoop-Core Performance Analysis Areas

Component	Metrics to Analyze Performance
Name Node	• Missing data blocks • Capacity remaining • Capacity used • Dead Data Nodes • Metric Endpoint • Volume failures total
Data Nodes	• Blocks: read, removed, replicated, written • Data read and write • Disk remaining • Failed volume • Metric Endpoints
YARN Resource Manager Metrics	• Node Manager active • Node Manager decommission, lost, rebooted, unhealthy • Application completed, failed, running • Metric Endpoints • Queue memory allocated, available • User active
YARN Node Manager Metrics	• Containers allocated, completed, failed, launched • Memory allocated, available, allocated ratio • Metric Endpoint • Virtual cores allocated, allocated ratio, available, total

Tools that provide performance monitoring of servers and networks used in a Hadoop cluster include Ganglia, Nagios, Cacti, Zabbix, and DataDog. Tools specific to Hadoop, such as Apache Ambari and Cloudera Manager require that the Hadoop cluster and all ecosystem components be deployed through their interface. Thus, these tools cannot be added into the environment once it is deployed.

A technique proposed to enhance the Hadoop AC system performance is to use hardware accelerators or trusted processor modules (TPMs). Several researchers have proposed the TPM technique to accelerate and increase the AC process security in Hadoop [91] [92] [93] .

Summary of BDP ABAC Research

There are various approaches to defining AC polices and attributes, with standards helping to achieve consistency. ABAC has been identified by many researchers as a leading method that can achieve fine grain AC in big data systems. Tracking the use of a provenance attribute is critical when privacy-preserving, sanitization algorithms are executed to lower the data sensitivity level. A layered architecture is needed for big data AC challenges based upon the surveyed technical approaches used to achieve big data fine grain AC. To achieve fine-grain AC in big data systems, security services need to extend beyond current approaches where big data AC is applied primarily at the boundary proxy server.

Based upon the proposed use case and mandatory governance policies, the key considerations for AC in big data Hadoop HPC environments are as follows:

- Define attributes for subjects, objects, and environmental conditions, including provenance, to track privacy-preserving algorithm execution
- Place AC services throughout the Hadoop ecosystem with distributed policy decision points, providing layered boundary and internal data node protections.
- Evaluate the AC approach from a functional performance perspective and based on a security services assessment against threats and using formal reviews.

Analyzing open problems through advanced AC research for big data systems will enable a continued expansion in the use of these sets while addressing privacy-preserving security requirements. By using ABAC, including provenance tracking, security services can be applied in the compute and store cluster. This approach will enable access to groups of less-privileged users.

CHAPTER FOUR: DESIGN OF EXPERIMENT

The experimental prototype we developed to analyze BDP security is described in this chapter. A key requirement was the generation of relevant data. Thus, our first area of development was to generate a representative data set with information at multiple sensitivity levels. We extended an open-source healthcare data generator to generate representative social media messages associated with the healthcare conditions.

We used this generated data to create a PySpark data processing lifecycle that executes on Apache Hadoop. Using Hadoop ecosystem security components, Apache Ranger and Apache Atlas we analyzed the security features and ability to support ABAC. An overview of our experiment design is provided in the following sections and additional details are available at our GitHub repository.

Experiment Design

Statement of Submission for Publication:

This work will be submitted for publication in the Wiley Journal: Security and Privacy.

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 compute 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 a broader 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 upon open-source software from the Hadoop ecosystem. The current default security configuration is primarily based upon security firewalls, proxy servers, and/or virtual private networks at the cluster boundary. These services are only recently integrating fine grain security services. Service providers are enhancing cloud BDP security based upon 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 security of BDP applications running in the cloud, we built an Apache Hadoop cluster with Spark parallel processing allow us full access to configuration and log files to achieve a deeper understanding of the security and performance implications. The focus of the experiments are the ABAC services for data stored in Hadoop and processed in parallel through Spark applications.

Environment Configuration Details

The BDP representation for the experiment consists of AWS EC2 Instances using the CentOS 7 Amazon Machine Image (AMI). The following open-source Apache software provided a realistic BDP representation: Hadoop version 2.10.1, 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 four Data Nodes. Two more instances were used for Ranger and Atlas. Free IPA was used for directory services (LDAP) on another EC2 instance, for a total of 10 EC2 instances. This configuration allowed us to isolate any potential performance issues and manage the configurations of the Hadoop ecosystem components independently. Plans are to apply performance analysis tools and Apache Knox for additional security experiments. This configuration is depicted in Figure 14.

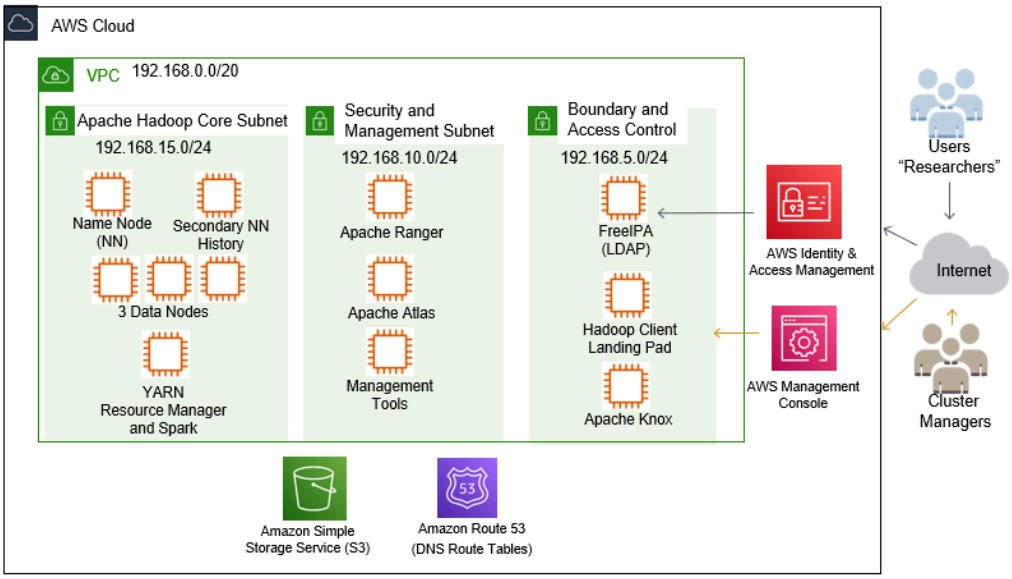


Figure 15: BDP Security Experiment Configuration

Experiment Execution Details

As shown in Figure 15, the represented BDP consists of six PySpark programs that process data sequentially. The PySpark programs and data sets used in this experiment are posted at Github[32]. Starting with two data sets from synthetic data generators, Synthea™ [33] and SynSocial[34], the data

[32] https://github.com/AnneMT/SEHadoop
[33] https://github.com/synthetichealth/synthea
[34] https://github.com/AnneMT/SynSocial

are filtered, joined, protected with a privacy preserving hash and encryption, and then finally analyzed to output the aggregated results. We believe this data processing sequence represents a realistic, yet simplified set of steps that require data security protections. Each of the data sets is stored in HDFS. Access control to the files is based on the HDFS POSIX-style file permission (read, write execute/user, group) however, with the Ranger HDFS plugin enabled on the Name Node, access decisions that are denied are referred to Ranger for evaluation. That is, HDFS file permissions dominate Ranger policies, and thus, if a user (e.g., local account or LDAP) on the HDFS Name Node has permissions (i.e., the user is defined as the file owner in the HDFS file permission settings), then access is granted. If the user does not have HDFS file/folder permissions, then Ranger is checked; whereas if Ranger has policies that permit access to the user, then access is granted.

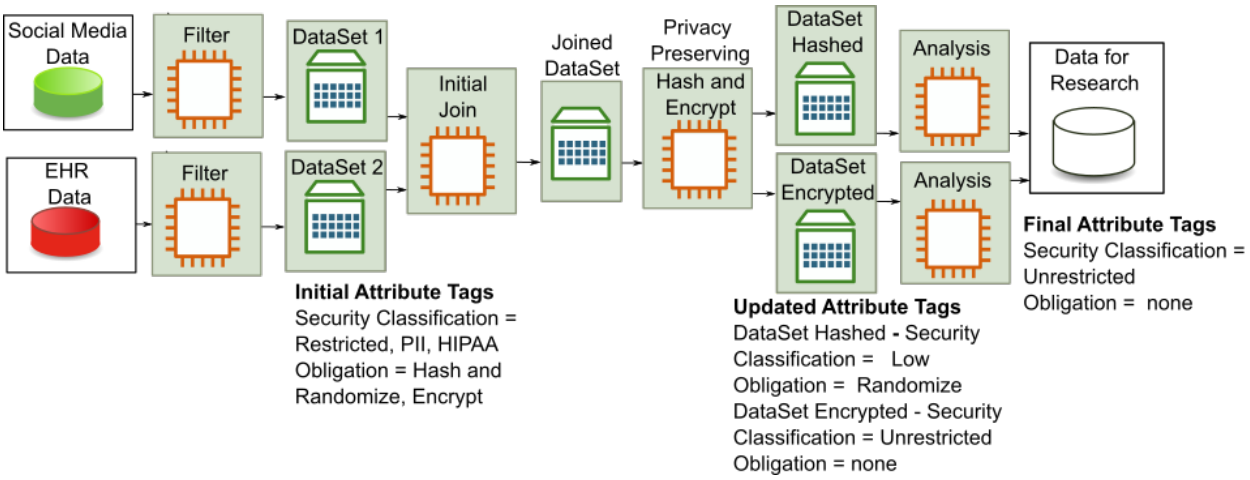


Figure 16: Represented Healthcare Use Case Data Lifecycle with Provenance Attributes

To force permissions to be determined by Ranger, the HDFS file permission settings have to be changed to eliminate access to all users (e.g., hdfs dfs -chmod -R 000 /user/foldername). Ranger is a framework for Hadoop ecosystem security in that it consists of a number of components that can optionally be installed. The core components are the administrative portal

and policy server. Security policies defined using the administrative portal are stored in a policy database (MySQL by default, or optionally, Oracle, Postgres or other database).

Policy details that are managed include resource maps based upon the ecosystem component, all internal and external users, roles, security zones, service details, tags, and synchronization status. Lightweight Java programs, i.e., Ranger plugins, are installed on Hadoop components, (e.g., HDFS NameNode, Hive Server, YARN, and Hbase Server) to periodically pull policies from the central Ranger server and store the policies locally using a REST API. When a user request comes to any Hadoop component, the plugin intercepts the request and evaluates it against the security policy to make a decision on whether the user request should be authorized. The risk of timing attacks that take advantage of out-of-sync policies will need to be evaluated and the synchronization frequency accordingly adjusted.

The Ranger Tag Sync service provides a critical role in making access control decisions based upon an ABAC model. The initial Ranger service was based upon a resource control model, and thus the tag sync service is a more recently added capability. This service enables synchronizing tags with Atlas and defining and managing authorization policies based upon these tags.

Atlas is also considered a framework of components that require considerable configuration. Once fully configured, it can achieve its role in configuring and managing security labels (e.g., attributes, tags, and classifications) and their relationships across the Hadoop ecosystem. It operates in a passive mode and depends upon the information sent to it from the ecosystem components. Thus, when new data, processes, or users are added to the ecosystem, “types” (which roughly correspond to object classes) need to be created, if not previously defined, and “entities” (which are basically instantiations of those object classes) need to be entered into the Atlas database. These definitions include not only the attributes for a specific entity, but also

relationships and tag inheritance, for example, a dataset can be identified as an input to a process and the output would then have the same attributes or classification. This provides management of the data lineage and propagation of the classification.

Curl commands input to the Atlas API maintain this information regarding the datasets. Administrators include these commands in scripts that run regularly and crawl through the data store. These scripts are referred to as "hooks," and there are some data stores in the Hadoop ecosystems where these hooks are available as part of the Atlas distribution.

The configuration and integration of these hooks requires an in-depth knowledge of data changes at the Hadoop ecosystem component and how to flow that information to Atlas. Therefore, without a proactive design and configuration of these hooks, there is a high likelihood that the data storage will become out of sync. As an example of a potential risk, if a rogue user of the data processing environment copies data into a folder or section of the HDFS that is not tracked and tagged in Atlas, the security policies will not be enforced by Ranger. This could cause exfiltration of the data in a manner that bypasses the intended security policies.

Atlas is a framework that requires significant developer participation in the glue code to bring all the component pieces together to form an integrated, complete security service.

Experiment Design Observations

During the setup of this prototype experiment, several observations were made in association with the potential limitations on achieving a secured BDP environment. These findings are in three main areas: integration and configuration, security confirmation or assertion, and the use of previous security best practices. A significant challenge with setting up the prototype is that the free and trial version of managed Hadoop configurations from Cloudera and Hortonworks are no longer available. Therefore, we used open-source Apache Hadoop. Significant expertise is

required to compile, create repositories, and integrate multiple Hadoop ecosystem components to use the Apache Ambari[35] configuration. We expect that many of the security features and issues may be hidden in the commercial, compiled versions of integrated software suites. Our focus for this initial analysis is on the security and performance findings with the core Apache Hadoop components. We believe these results would form a basis for security services analysis on more complex managed systems.

A large number of integration and configuration options provide great technical flexibility in the Hadoop ecosystem; however, it also presents a significant security risk. For example, each Ranger component has its own configuration file (install.properties) that must be set with a number of component and service specific variables. An incorrect configuration of any of these files can result in security policy bypasses.

Ranger and Atlas both provide a passive role in the Hadoop ecosystem. Rather than actively scanning or otherwise interacting with Hadoop components to confirm the synchronization of the configurations and provide assertive feedback on the system status. Correct operation relies upon API or hook configurations that require significant system-specific knowledge and programming skills.

Hadoop is not designed to take advantage of previous security capabilities and design methodologies. Flexibility results in complexity, which traditionally results in security risk. For example, the use of SELinux on systems that comprise the Hadoop ecosystem should be able to provide high granularity in controlling the processes executed by the Linux kernel; however, the larger the number of components, tools, and other applications that are used, the larger the number of corresponding SELinux policies that are required. All of the component installation

[35] https://ambari.apache.org/

directions we have reviewed recommend setting SELinux to “disabled” mode. Even the process of developing SELinux policies using “permissive” mode is viewed as a complex, tedious process that requires frequent revisions every time a new component or feature is added to the ecosystem [94]. Although research has been conducted in this area, incorporating more rigorous process controls is still needed to achieve multi-level security [95].

In Chapter 5, we describe the results of executing the experiment with a large data set subject to different security policies, including the performance information. The generation of this large data set that was stored in Hadoop and processed using Spark as part of this experiment is described in the following section.

Data Generation

Statement of Prior Publication:

This work was previously published at the I/ITSEC 2020 conference, as listed in Reference 96.

We developed a synthetic social media data generator, that we titled “SynSocial” [96]. It is based upon and linked to synthetic medical data generated by an Electronic Health Record (EHR) data generator, such as Synthea™ [47]. The approach to generating the data was designed as an initial open-source framework that could be expanded to generate social media data that is relevant and related to medical conditions and treatments over time. This type of generated social media data synthesized with medical healthcare data is needed for a variety of test and research applications, such as the detection and spread of diseases, early detection of illness or the effectiveness of behavior modification programs to improve health.

Like a recommender system in reverse, SynSocial generates social media data, (e.g., Tweets), based upon health care information. Previous research using recommender systems based upon real-world social media data has predicted a number of medical conditions, including flu

outbreak trends [97] [98] [99] [100]. Making connected synthetic social-media and healthcare data available to researchers enables the investigation of new algorithms and model development with open data that is free from security or proprietary controls. Current research is incorporating factors such as demographic and community information, such as age, sex, and relationships (friends, client/patient) in social media data as predictors of medical conditions [101] [102] [103]. The proposed synthetic social media data generator also incorporates these dimensions in the design. In this paper, we focus on Twitter data generation in this initial framework development.

Currently, several commercial entities are selling Twitter data analysis services. Twitter publishes some statistics about its usage. The motivation is to attract paid advertisers and not necessarily researchers, so the heuristic values are limited. However, Twitter provides an API and data can be scraped from this interface for a variety of research purposes. A challenge with using real-world social media data is the potential to disclose sensitive or personal information, especially when that data is connected with medical conditions.

Motivation

Public social media data sharing has been shown to be a new source of information to identify and analyze a variety of issues. Public health concerns and trends, in particular, have been identified by researchers as an area where new insights can be gained from social media data. Twitter is a leading source for this type of information [104] [105] [106] [107].

However, to realize the potential of these insights, open, unsensitive information, free from Personally Identifiable Information (PII) is needed to develop algorithms and experiment with security features. For example, anonymization algorithms and residual risk of re-identification through inference could be tested using synthetic data. The challenge of developing synthetic

social media data and anonymized data sets obtained through social media system APIs requires research [108]. This project contributes to these efforts by proposing a method to generate synthetic social media data, specifically Twitter data, that is connected to synthetic medical data. These generated data sets can then be used to develop models and conduct analysis without concern about protecting PII and healthcare data as mandated by the Health Insurance Portability and Accountability Act (HIPAA), U.S. legislation that requires data privacy and security to safeguard certain medical information.

Novel Contributions

The unique and novel contribution for this data generator, SynSocial[36] is the connection of synthetic medical information to synthetically generated Twitter data. The medical information is connected to the social media data over a patient's lifetime. The generated messages are produced at a rate that is viewed as realistic based upon the age of the patient and their associated medical conditions. For example, the types of messages generated and the rate that they are generated vary over the lifetime of the patient. The data generator has been designed to enable adding higher levels of fidelity and realism as needed for various research objectives. The approach for validating the data produced by SynSocial was done by analyzing the frequency and quantity of messages when users are healthy (baseline) and when under medical conditions. This initial capability considers many factors influencing social media data generation and provides a framework that can be easily extended by others.

[36] Code is available at https://github.com/AnneMT/SynSocial

System Description - Data Generator Design

In this section, we describe the design of SynSocial, our patient social media data generator, based on one particular EHR data generator. However, our design is generic and can be easily modified to be used with other EHR data generators.

The EHR data generator used, Synthea™, is an open-source synthetic EHR data generator that incorporates a wide variety of diseases [47][37]. Initially, the top ten reasons to visit a Primary Care Physician (PCP) and the top ten diseases that cause loss of life, ("Two Top Tens"), were used to start the simulator project. The project has been expanded and currently has over 90 different modules that generate data on a wide variety of diseases.

The synthetic EHR include a number of states starting with "Condition Onset" and transitioning through various states associated with the disease progression, and then ending with "Terminal," states that result in generation of an Electronic Health Record (EHR). For SynSocial, the generation of medically related social media messages starts with condition-onset and ends at condition-abatement. If a medical condition end date is not included in the EHR data, an assumed date of one year after onset is used, based upon the idea that the number of social media messages written after a persistent long-term condition would drop-off after that period of time. In SynSocial, birth and death dates are also used as input to the start and end of the social media message generation dates. For example, social media message generation starts at the age of 18. The number of messages generated decreases as the patients age and the contents of the messages corresponds to the patient's age group. Medical states such as prescribed medications and lab results included in the EHR are incorporated to expand the generation of medical-related social media messages.

[37] Available at https://syntheticahealth.github.io/synthea/

Overall Design

The overall design of SynSocial is to enable the generation of a large volume of data in parallel. Each generated Tweet is appended as a JSON formatted message to the output file. Currently, modifications or deletions of previously generated data is not incorporated into the data generator design but could be added in the future. Information such as marital status, education level, and other factors that influence a person's social media behavior could also be incorporated in the future. SynSocial uses a baseline Tweet generation rate based upon age and combines that with the generation of medical-related messages with the occurrence of a single or multiple conditions, as listed in the generated corresponding EHR data. The rate and contents of the generated Tweets is modified based upon combining and deconflicting the severity of the co-occurring conditions and baseline rate. Figure 16 highlights the program actions and interconnection. The logical flow of the SynSocial media data generator considers the message generation rate over the patient's lifetime and varies the rate and type of messages posted based upon the conditions and an age-associated baseline rate.

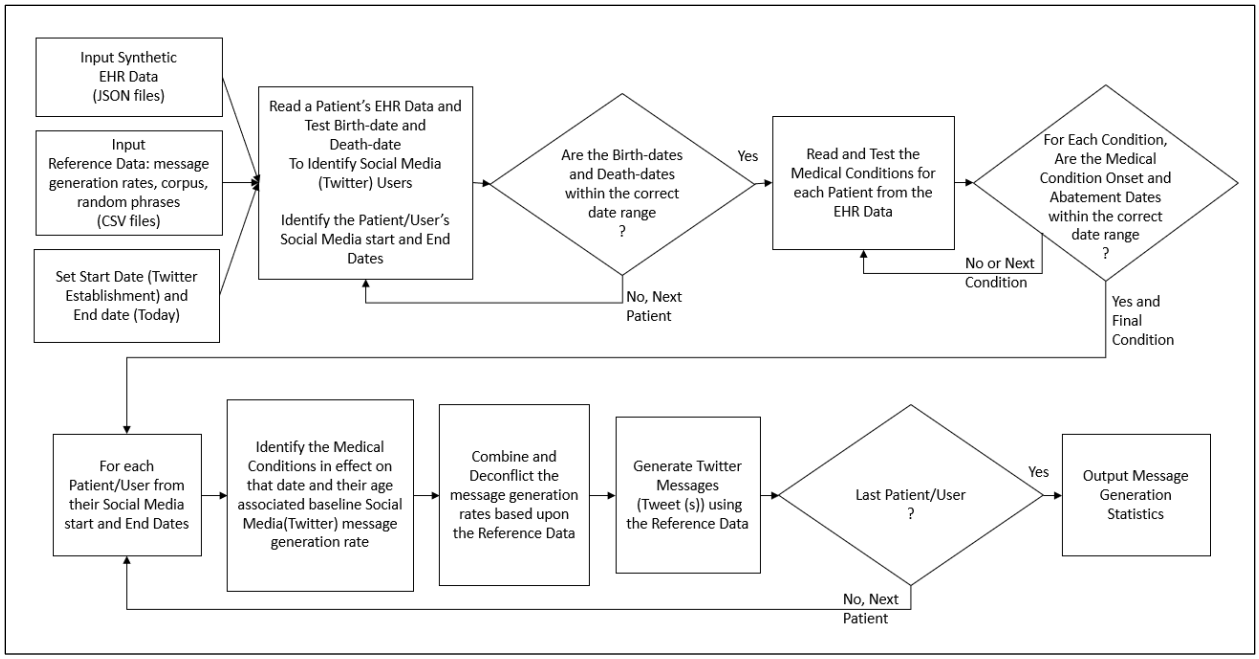


Figure 17: Overall SynSocial Social Media Data Generator Design

Data Input From Synthetic EHR Medical Data Generator

Synthetic medical data generators can be configured to produce a data set for a specified population size and support a variety of different configuration items, such as the locality of the population represented, and the format of the output data. SynSocial was designed to use as input JSON formatted files that contain each individual patient's medical condition over their lifetime. An important field in the generated EHR data is the SNOMED-CT medical condition code. This is the standard language for encoding medical terms and conditions that has evolved over many years into international adoption. The translation of the codes into the medical condition is available online from SNOMED[38] and other sources such as the U.S. National Institute of Health (NIH)[39]. Short text messages based upon the SNOMED-CT codes are used to generate the full social media message (i.e., Tweet). For example, Table 9 lists example Tweets for the common

[38] https://www.snomed.org
[39] https://www.nlm.nih.gov/healthit/snomedct/

disorder sinusitis. This corpus of message text is provided as a look-up file where the message is randomly selected from the phrases associated with the medical condition code. The original basis of the corpus is Twitter itself, however, none of the messages are an exact duplicate of real-world Twitter messages. The real-world user mentions and replies to real-world handles (i.e., use of "@Twitter username") has been removed and random phrases before and after message texts are added.

Table 9. Example Synthetic Social Media Contents

1	That was one of the worst cases of sinusitis I've had in a long time.
2	I have sinusitis. Any tips on getting rid of it? I would like to avoid antibiotics if possible.
3	What do you guys swear by for your allergies, esp. if you suffer from sinusitis?
4	Warm water Lemon, Ginger, Garlic, Turmeric and Cayenne pepper mix has helped me so much.
5	Sore eyes, tonsillitis and sinusitis, Wow
6	Currently suffering from sinusitis.
7	Can I get a new nose? This sinusitis got me good
8	Raging case of sinusitis.
9	Flu into a cold now acute sinusitis
10	Still feeling terrible and suffering from sinusitis

Data Output Format

The generated data is output in a JSON format based upon the messages that can be extracted from the Twitter developer interface API[40]. Not all fields are populated, however this can be expanded to incorporate additional complex dynamics in social media, (i.e., followers/following communities of interest). Tweets are the basic atomic building blocks that are posted, liked or reposted on Twitter. Tweets are also known as "status updates." The information contained in the

[40] https://developer.twitter.com/en/docs/tweets/data-dictionary/guides/tweet-timeline

tuple is based upon the information and format specified by the Twitter API. An example of the output social media message, in Twitter's Tweet JSON format is shown in Figure 17.


```
 {"created_at": "Mon Jan 01 08:38:22 +0000 2007",
 "id_str": "10101083822200710221562461",
 "text": "absolutely, consulting a doctor regarding my chronic sinus
condition, let's chat later",
 "user": {
      "id": 200511085042678197,
      "id_str": "200511085042678197",
      "name": "Tonja658",
      "screen_name": "@Tonja6fishnet",
      "location": null},
 "place": {"country": "United States", "name": "Palmer Town, Massachusetts"},
 "entities": {"hashtags": [], "urls": []},
 "extended_entities": {"media": []}}
```


Figure 18: Example SynSocial Social Media Generated Message

Key fields generated by SynSocial are:

- Time stamp, (created_at), when the Tweet was published (created at date), with the time randomly generated
- Message content, (text), of the Tweet, randomly selected from the baseline age-correlated and medical condition corpus reference file
- Geographic location, (place), based upon the address in the EHR data file
- Author of the Tweet, (user screen_name), derived from the SynSocial generated name

This format includes the primary fields from a Twitter data object and represents data that is likely curated to analyze messages associated with medical conditions. A full Tweet contains additional fields that could be populated by SynSocial to meet a variety of research requirements. To match the emergence, use and enhancements of Twitter, the earliest date (time stamp) used is January 1, 2007 since Twitter didn't exist until 2006 and some features were created later, such as geo tagging.

Twitter Handle-Name Generation

SynSocial uses the first name generated by the EHR data generator combined with a random word (noun) appended to create a Twitter nickname or handle. The random word is currently chosen from a reference file, however, to increase the diversity, the words could also be created using a random word or name generation tool[41]. The desired level of realism would influence the source (dictionary or random name generation tool).

Twitter Message Generation Model

Our SynSocial generator creates two types of Twitter messages (Tweets): normal messages, and medical-condition messages. Normal messages are the general tweet messages generated by users that are not related to their medical conditions; medical-condition messages are tweet messages where the users talk about or discuss their current medical conditions and health concerns.

For each type of Twitter messages, the number of messages generated by a user per day will be modeled to follow Poisson Distribution $X \sim Pois(\lambda)$ where the rate λ is the mean value of the number of messages generated per day. At any given time, a user may be healthy, or may have one or more medical conditions. Let us denote the number of Twitter messages generated by a particular user in a day is *N*, then Equation (8) is defined as:

$$N = \alpha \cdot N_h + N_m \quad (8)$$

Where N_h is the number of normal Twitter messages when the user is healthy (called 'baseline' messages) and N_m is the number of medical-condition Twitter messages. $N_h \sim Pois(\gamma)$ where γ is the rate of baseline messages. $N_m \sim Pois(\lambda)$ where λ is the rate medical-condition Twitter messages are generated. When the user is healthy without any medical conditions, λ would be 0.

[41] https://randomwordgenerator.com/name.php

In Equation (8), the important parameter, α ($\alpha \in [0,1]$), represents the *illness impact* to a user's daily normal Twitter message generation: when a user is sick with one or multiple medical conditions, the user would reduce their normal Tweets, but will generate some messages related to the illness, expressing their feelings, comments, or concerns towards their current medical conditions. If a user is seriously sick, such as staying in hospital, α could be as small as 0 meaning that the user has no ability to generate normal Twitter messages due to this medical condition.

Suppose there are *n* medical conditions in the generated EHR data. For each medical condition SNOMED-CT code i, $(i = 1, 2, \cdots n)$, we define in SynSocial the corresponding illness impact factor α_i, and medical-condition Tweet Poisson distribution rate λ_i. If the user has one and only one medical condition i , the user's generated medical-condition tweet rate is simply $\lambda = \lambda_i$, and α in Equation (1) is simply equal to α_i.

If the user has multiple illnesses, (i.e., the user has a set of medical conditions $\mathcal{S}$), the user's daily generated Twitter messages would be modeled by Equation (1) with the following parameters, Equation (9):

$$\alpha = \min_{i \in \mathcal{S}} \alpha_i \text{ and } \lambda = \sum_{i \in \mathcal{S}} \lambda_i. \quad (9)$$

Table 10 shows the parameters used for the normal social Tweet messages generated by our SynSocial program. For normal Tweet message generation, we classify users based on their age group. γ is the Poisson distribution rate, i.e., the average number of normal Tweet messages generated by a user per day. The daily tweet rate shown in Table 10 are set in an Excel configuration file in SynSocial; so, they can easily be changed to support different research objectives. Example topics contained in the corpus of message texts are also listed. The actual

message is randomly selected from an Excel file and then a random phrase is appended to the beginning and end of the message.

Table 10. Baseline Message Generation Rate

Age (years)	Daily Normal Tweet Rate (γ)	Example Tweet Text Topics
Birth to 18	0	nil
18 to 20	4	college, dating, job search, first job, working, wedding, first home
20 to 25	4	college, dating, job search, first job, working, wedding, first home
25 to 30	4	parties, children child care, food, vacation, sports, job change
30 to 40	3	children, moving, job, promotion, social activities
40 to 50	3	food, vacation, sports, hobbies
50 to 60	2	grown children, grandchildren, moving, job, social activities
60 to 70	2	retirement, vacations, home repair, hobbies
70 to 85	1	retirement, travel, home repair, hobbies
85 to 99	1	travel, home repair, hobbies
99 or older	0	nil

SynSocial is flexible and open so that alternative approaches to creating the corpus of message text can be used. For example, a random text or phrase generator[42] could be used to create the message corpus. Example values of the medical-condition Tweet generation parameters, λ and α, for several SNOMED-CT codes is listed in Table 11.

Table 11. Example Medical Condition Message Generation Rates and Severity

SNOMED-CT Code	Condition	Daily Medical Condition Message Generation Rate (λ)	Impact on the Baseline Rate (α)
4448144009	Viral Sinusitis	1	0.25
162864005	Obesity	2	1
40275004	Contact Dermatitis	1	0.5
72892002	Normal Pregnancy	2	0.5
198992004	Antepartum Eclampsia	1	0

[42] such as http://theidiomatic.com/

For this example, the medical condition code 198992004, Antepartum Eclampsia, is considered a severe medical condition, which would dramatically impact the patient's normal social media post activity, making the normal post to be zero. In such situation, the patient would be posting a small number of messages only about their severe conditions and not about other topics.

Related Work

Researchers have proposed several alternatives for social media data generation. Specifically, Yu et al. proposed the BSMA-GEN simulation, [109]. This effort addressed the need for parallel execution and scaling to produce a large data set. The contributions also included ensuring the produced data is in a realistic format and addresses behaviors such as re-Tweeting.

Sagduyu, Grushin and Shi proposed a synthetic social media data generator that uses a novel concept in generating synthetic graphs to realistically address who is talking to whom [108]. However, a challenge in applying this concept to Twitter, is that the media operates as a broadcast to many followers rather than a direct person to person exchange. This research effort also addressed synthetic text generation using innovative approaches such as chat-bots or social media bots. They qualified the utility of this strategy through human experiments to measure the realism of the synthetically generated messages. Overall, they were able to produce texts that are grammatically correct and coherent.

Another important area that has been researched is geographic tag (geo-tag) references in messages and their relevance in studying various issues. For example, Sadilek, Kautz, and Silenzio examined disease transmission using a combination of social media posts and associated geo-tag information [103]. Moreira, Tiago, and Pianho used geo-tags in combination with social

media data to examine emotions and stress in smart cities [110]. Indicators of mental health issues in social media data, as proposed by Yazdavar et al., is an interesting emerging area that could provide great insights, but also contain many PII and HIPAA sensitivities [111]. Nugyen et al. all examined food-related illnesses using social media posts and associated geo-tag data [112]. These innovative research efforts are providing opportunities to gain greater insights into a number of health issues based upon real-world data sets. Synthea has been recently updated to include COVID-19 medical conditions, so testing algorithms that track their geographic area using geo-tagged Tweets might be an interesting area for investigation.

To further these types of research efforts, models and simulations can be developed, tested and enhanced using synthetically generated social media data connected to healthcare information as proposed in this paper. Future efforts to advance the initial concepts proposed would be to conduct validation and verification of the generated messages for frequency and relevance against actual real-world data, (i.e., compare the generated Tweets to real-world Tweets). The realism could be enhanced using additional rich formats and data types, (e.g., pictures, web-site links, hashtags, mentions, images, videos, clips, and sound). Another area for future investigation could be modeling the behavior and impact of social events and influencers as input to text generation. This may enable the analysis of using social media to encourage more healthy behaviors, such as exercising and eating healthy foods.

Data Generator Evaluation

Evaluation of the data produced by SynSocial was considered based upon the quantity and quality of the data generated. The publicly available statistics on the number of real-world Twitter messages generated by age and medical conditions were identified. For example, the prevalence of illness related Twitter messages has been identified by researchers [98]. In Figure

18, an example of the number of generated baseline and medical condition social media messages is shown over multiple years. The occurrence of medical conditions is indicated, with the impact of a severe medical condition, Antepartum Enclampsia, on the suppression of baseline messages highlighted. Between 2016 and 2017 this synthetic patient did not have any medical conditions, and the messages are generated at a rate that matches a Poisson Distribution for a mean of 3 which corresponds to the age bracket (30 to 40) of this patient during this time, as listed in Table 10. The 2011 to 2016 timeframe was omitted (as noted by the jagged line) to show more of the medical timeline for this example social media user/patient.

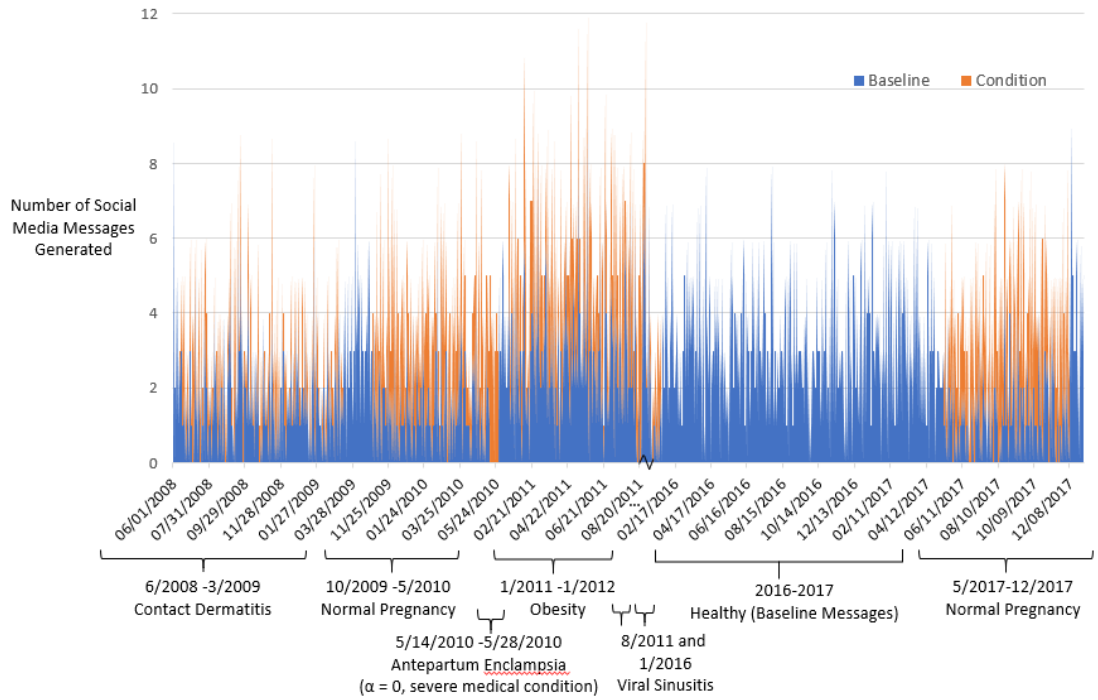


Figure 19: Example Social Media Data Generation for an Individual User/Patient

To examine the qualitative aspects of the generated messages, whether the contents of the messages correspond to what people Tweet about was considered. Less is published about trends in message contents as it relates to medical conditions. The overall goal was to ensure that the generated data contained no confidential or sensitive data. The message body corpus was based

upon real-world data, however Twitter "mentions" which reference real user names were removed and additional random data was added to generate synthetic data. The process of extracting a baseline and medical condition corpus of messages could be further automated to increase the anonymization, scrambling of a larger set of real-world messages used as the base for the message generation. The size of the generated data is summarized in Table 12 below. This makes this synthetic data well suited for big data experiments that require representative data that has both a high degree of sensitivity, (such as EHR data), and open unclassified information, (such as social media data).

Table 12. Example Social Media Data Generation Size

Input Patient's EHR Data File Size	Number of Medical Conditions in Input Data	Output User's Social Media Message File Size	Number of Medical Condition Messages (Tweets)	Number of Baseline Messages (Tweets)
718 KB	10	7,413 KB	1,791	12,818

There are several research projects [113] [109] that focus on understanding the relationships between social media users, e.g, who is following who, numbers of likes. This type of research could be incorporated into SynSocial as mentions (e.g., using an "@" tag). Also, there is research [114] associated with understanding the sentiment of messages which could be used in building the corpus of the referenced message sets. Artificial generation of message text to portray conversations is also an area of research that could influence the design, however the nature of Twitter, is more akin to a micro-blog post rather than a conversation, so a Twitter chat-bot[43]

[43] https://marketing.twitter.com/emea/en_gb/insights/how-to-plan-and-analyse-a-twitter-chatbot

might be applied to represent comments on messages. These are areas for potential further research, development and expansion of SynSocial.

Future Data Generation Work

We proposed SynSocial social message data generator that is connected to synthetically generated medical data from an open source medical data generator. SynSocial provides a useful resource for developers experimenting with new insights that can be gained from social media data without concerns for PII and HIPAA requirements. The initial framework of the design allows for it to be extended for higher fidelity realism as needed for a larger number of complex medical conditions. This is a first, unique effort to provide the tools necessary to advance large scale data analysis from two previously unconnected sources, one very sensitive (healthcare data) and the other open public (social media data), thus enabling the development of new data analysis capabilities.

CHAPTER FIVE: EXPERIMENT EVALUATION

Statement of Submission for Publication:

This work will be submitted for publication in the ACM Journal: DTRAP.

The focus of our prototype and experiments is evaluating AC, specifically achieving ABAC, as described in this chapter. Active research in BDP security currently covers a wide variety of issues, including homomorphic encryption [115] and developing secure applications that can quickly become operational (DevSecOps) [116]. These advances depend on reliable AC. As Big Data continues to grow and migrate to the Cloud, there is a need for multi-tenant, multi-sensitivity cluster computing depends upon fine-grained AC. We approached this evaluation from a perspective of vulnerabilities to cybersecurity attacks, ability to consistently employ security services to thwart attacks and performance impacts when security services are employed.

To protect data in BDP environments, the system security managers and administrators need to think and test each ecosystem component and the collective environment like a cybersecurity attacker. Many security features have been added to BDP systems, such as identification using Kerberos and encryption of data at rest, however a comprehensive system security perspective also needs to be applied with these individual capabilities. Attackers take advantage of any open port, protocol, service in the compute, store, and communication environment where they can gain a foothold to introduce malware or exfiltrate data. Confidentiality, integrity, and availability need to be applied in a consistent, well integrated, layered manner across the environment.

Experiment Evaluation Methodology

There are a variety of alternative approaches to AC under research [117]. The security of cloud based BDP, such as AWS Elastic Map Reduce (EMR), Google Cloud Platform (GCP) and

Microsoft Azure, is largely based upon perimeter, "castle-wall" strategies. There is an assumption that everyone with access to the distributed, cluster computing capability has full access to all data. Fine grain access controls that separate data access based upon authorizations, roles, and/or data sensitivity are not configured by default in automated cloud-based clusters. Additionally, there is no mechanism to verify security features are implemented, configured, and providing necessary separation of duty 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 wide variety of users with different analytic processing requirements.

For example, with Kerberos, the service ticket issued after authentication (Authentication Service) by the Ticket Granting Server (TGS) is valid for a period of time. Hijacking credentials to generate Kerberos "golden" tickets that never expire is an attack vector that has been exploited in publicly disclosed attacks, such as the Sony attack [118] [119]. The threat of this type of attack needs to be carefully assessed. It should not be assumed that attackers cannot move beyond the perimeter boundary [120] [121].

Our methodology for evaluating the ability to achieve ABAC in our experimental prototype was to examine AC services throughout the cluster, identify potential attacks to the AC and related security services, and assess the ability to configure the ecosystem components to provide AC using flexible, dynamic attribute definitions.

Apache Hadoop Background Details

The core Apache Hadoop components and their interaction is shown in Figure 19. 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 Manger tracks the execution of data

processing through the distrubed Node Mangers. The Resource Manager is a component of the Hadoop Yet Another Resource Negotiator (YARN) framework, introduced in version 2. Each Node Manager launches Application Masters (AppMasters) that allocate Java Virtual Machine (JVM) containers for parrallel process execution. The Job History server and Application Manger oversee job execution across clusters. This architecture enables distributed processing and storage on commondity machines using a programming models, such as MapReduce (which is part of the core-Hadoop distribution), Apache Spark, and other independetly developed open source tools.

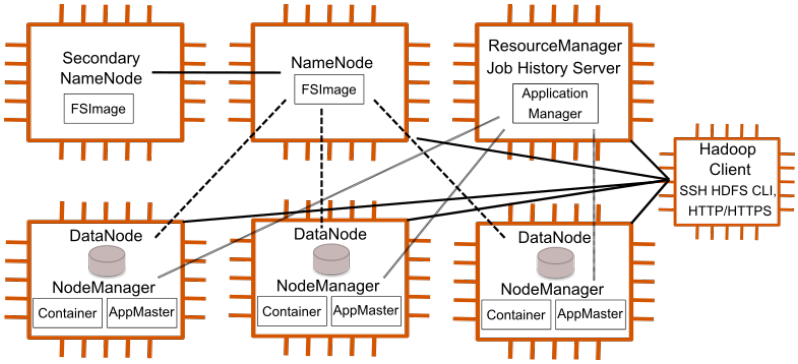


Figure 20: HDFS, YARN, and Hadoop Client Component Interfaces

During a Hadoop Distributed File System (HDFS) read or write, the Hadoop client interfaces to the core Hadoop components using Secure Shell (SSH) and HDFS command line interface (CLI), a web browser using HTTP / HTTPS, or other custom TCP interface, such as Java Database Connectivity (JDBC). The Name Node provides the client the location for reading or is assigned the location for writing data blocks that comprise the data file. Then the client interfaces directly to each Data Node to read and write the file data blocks. With the client applications used by data analysts interfacing with all Data Nodes, Name Node and Resource manager, there is potentially a very broad 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, the Application Manager, interfaces with each Node Manger co-located with the Data Nodes to initiate an Application Master (AppMaster) which then coordinates Data Node resources (e.g., memory, processors) in JVM containers across the cluster.

The design is for Hadoop clients to communicate with every component in the Hadoop core architecture, except for the Secondary Name Node. The Secondary Name 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 to increase its availability Name Node if the FSImage file is corrupted.

From a cybersecurity perspective, the ability to interface with every Data Node, the Name Node and the Resource Manger provides Hadoop clients a broad access surface. This could also potentially result in significant issues if the appropriate access controls are not in place.

New higher reliability Hadoop configurations, such as the Name Node Federation and High Availability configurations, further expand the client to component interfaces. Multiple copies of files and multiple methods for submitting jobs into the ecosystem further complicate the security of these more advanced Hadoop configurations.

The flexibility in the Hadoop framework, such as the exposed application programming interfaces (APIs) enabling 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 these projects enable Structured Query Language (SQL) and SQL-like

interfaces, data ingesting services, job scheduling, provisioning, and management. These various capabilities offer easy entry points for data analysts.

There is active contribution and use of the open-source Apache Hadoop ecosystem by a wide variety of international organizations, such as universities, social media companies (Facebook), and cloud service providers (RackSpace, Amazon). Cloudera is a commercial company that is a leading provider of Hadoop support [122].

Areas of Investigation

In this section we discuss our approach and areas of analysis. We identify cybersecurity attacks specific to Apache Hadoop and related BDP technologies. This intends to inform further research in evaluating the residual attack surface after security services are in place and the need for additional protections.

A summary of the areas analyzed, and the potential attacks are listed in Table 13 below. The vulnerabilities are paths of entry into the system, such as unsecured communication protocols, open connections, and configurations that could be exploited. The cybersecurity attacks are examples of actual attacker actions and campaigns that have exploited the vulnerabilities. The attack campaigns are known frameworks and libraries of exploitation tools, such as Metasploit, Cobalt Strike, and others detailed on the ATT&CK web site in the techniques and software sections. Researchers have published reports that identifying these vulnerabilities on thousands of Internet-connected Hadoop systems [123] [124] [125] [126].

Table 13: Hadoop AC Service Areas, Vulnerabilities and Cybersecurity Threats

Access Control (AC) Areas	Existing Vulnerabilities	Cybersecurity Attacks
Hadoop File System (HDFS)	File system ACs are turned off by default and multiple ways they can be configured	Unauthorized reading and writing to HDFS folders
	SSL/TLS protections on the HDFS WebUI is 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, using NMap, OpenVAS and other port scanners identify open ports, protocols and services leading to Unauthorized connections and unauthorized 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 encrypts and deletes 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

Access Control (AC) Areas	Existing Vulnerabilities	Cybersecurity Attacks
Management Controls	Different management consoles to multiple ecosystem tools using various ports and login credentials	Attacks are undetected due to insufficient management visibility 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.	

Hadoop File Distributed File System (HDFS) Access Control (AC)

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

In HDFS the permission settings for files and folders is based upon the POSIX model. However, there are some differences since there is not a concept of executable files, i.e., program files are stored outside of HDFS. If HDFS is configured to conduct a permissions check for a file or directory accessed by a client process, the traditional, owner, group, other permission checking is tested. If a file permissions check fails, the client operation fails, whether that 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, then only the account Hadoop is run under, e.g., the hadoop superuser account, can access the files and directories. In this situation, proxy servers are typically employed to map all permitted users to more privileged superuser.

The contents of the file system and status of the block replication in the data nodes is displayed through a web-browser graphical user interface (WebUI). This interface and all the HDFS file reads and writes can execute over 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. Without the security settings enabled a wide variety of unauthorized data reads and writes could occur.

Although the basic capabilities to manage and control the HDFS are in place and based upon the familiar POSIX format, they are not enabled by default. The lack of default security configurations can result in many AC mistakes.

Operating System and Directory Access Control

Apache Hadoop user authentication and AC can occur at the Operating System (OS) (Linux) level or over the network by a directory server using LDAP, if configured. As described in the previous section, this is not configured by default, and could result in AC errors.

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

Without the host-based firewalls enabled, hosts can be subject to reconnaissance scanning by open-source tools such as Network Mapper (NMap) and OpenVAS. Reconnaissance provides attackers indicators of what is enabled. They can then derive potential weakness in hosted

software for further exploitation and attack e.g., Christmas Tree attacks. Also, flooding listening ports with active protocols can causes a denial-of-service situation.

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

Resource Management, YARN Access Controls

The Resource Manager, introduced in Hadoop 2.0 is a component of YARN, which stands for Yet Another Resource Negotiator. It oversees the division of processing load on Data Nodes using the Node Manager, Application Masters and Container daemons.

YARN obeys the HDFS file permissions settings, using the identity of the user (by default) for interacting with the file system. YARN considers only the files that the user owns for executing YARN program and does not perform any privileged action. It is possible however to specify a different user, so the YARN Resource Loader interacts with HDFS using that user's rights. When Hadoop is accessed by multiple users, it is recommended to create separate Resource Loader instances (one per user) instead of assigning additional permissions or groups to one user. This ensure HDFS ACs are 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.

For controlling who has permission to submit jobs to YARN, ACs can be configured in the context of job submission scheduling queues and pools. For example, when Spark users submit their jobs, they are added to job scheduler after authentication and AC decisions. YARN has three types of schedulers: capacity scheduler (default), first in first out (FIFO), and 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, flexible approach to managing job scheduling ACs, rather than configuring XML files, is for system administrators to use Apache Ranger. There are Apache Ranger YARN plugins that manage authorization policies on which users are allowed to submit jobs to YARN. These policies enforce who can submit to the YARN queue.

Without configurating YARN scheduler AC, the cluster can be subject to a wide variety of unauthorized job submissions. There are reports of several unauthorized cryptocurrency mining programs being detected in Hadoop clusters, such as DemonBot. A more destructive threat is ransomware which could destroy a large-scale cluster data set by encrypting the data and withholding the decryption key [127] [128] [129].

Service Level Authorizations

In BDP environments, client services need to incorporate identification, authentication, access control, and encrypt data exchanges. This applies to all actions including submitting jobs (Apache Spark and MapReduce), reading and writing files, and interacting with the web 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 health

status to the Name Node. Data Nodes also use an additional TCP/IP data transfer protocol to exchange data blocks [130].

Client applications, which for the Hadoop ecosystem include Apache HUE and Zeppelin, as well as the HDFS CLI, can incorporate security services. If configured, Kerberos enforces client authentications (identification). Data Nodes use SASL for authentication when using the data transfer protocol to exchange data blocks, if configured.

Authorizations are defined based on the Access Control Lists (ACL) contained in the hadoop-policy.xml configuration file, or as defined in an ecosystem component external to the core Hadoop components, i.e., Apache Ranger, as described in the following sections.

If configured, Secure Socket Layer (SSL)/Transport Layer Security (TLS) secures information exchanges. For Hadoop clients and servers, this protects RPCs. SSL/TLS also protects the web interface used by servers to expose their status; in that it can be configured with Hyper Text Transfer Protocol - Secure (HTTPS). This includes, for example, the Name Node and Resource Manger Web GUI.

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

Management of ACs for these various protocols, that are used by different components, are contained in configuration files. These can be very complex and difficult to translate to modern AC models such as ABAC. Use of a management console such as Apache Ranger is helps to centralize this configuration and ensure consistency. This is instrumental in achieving security objectives.

If not secured, eternally exposed protocols, HTTP, RPC, and JDBC-SQL, even when running on a high port range, can be subject to a wide variety of attacks. For example, web application attacks include SQL command injection, Cross Site Scripting (XSS), web shell implants, and brute force command injection.

Management Consoles

The complexity of integrating different BDP ecosystem components has been handled by management consoles such as Apache Ambari. However, they largely don't directly address security configurations. Most of the Hadoop consoles focus on performance statistics. Separate, independent management interfaces, such as Apache Ranger, have been developed to fill this role. Without a central view of the security status of the Hadoop, the system security managers can be blind to attacks. Detection of unauthorized actions may only occur after review of complex log files. Amazon has recently announced that EMR will integrate with Apache Ranger [131].

A common documented weakness (CWE-778) is that when security-critical events are not logged properly, detecting, and hindering malicious behavior is much more difficult and may hinder forensic analysis after an attack succeeds.

In Figure 20, the use of Apache Ranger and Apache Atlas to enforce and manage security policies with the core Hadoop components is shown. Interconnection of a directory server to manage user authentication and AC is also included. The following sections highlight the key capabilities of these additional tools support security management.

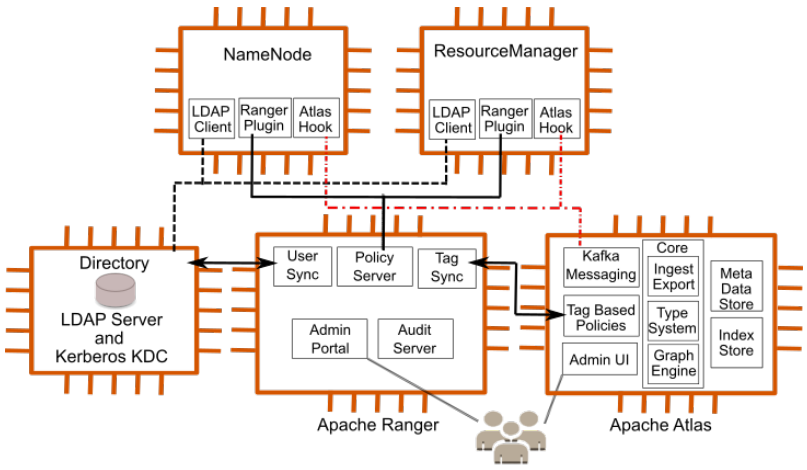


Figure 21: Apache Ranger and Apache Atlas Interfaces to HDFS Name Node and YARN Resource Manager

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 interface to administer security settings, including fine-grained authorizations (AC) in Hadoop components such as Hive, HDFS, and YARN. It helps achieve centralized audit tracking for each user action. The Ranger components include an administrative portal, policy storage (in MySQL or other database), audit log storage (in Solr, HDFS, or other database), a synchronization with user accounts in a directory (User Sync), a synchronization with metadata tags in Apache Atlas (Tag Sync), and Hadoop ecosystem component plugins.

Plugins are lightweight Java programs installed on Hadoop components (Hive Server, HDFS NameNode, YARN Resource Manager, etc.) to periodically pull policies from the central Ranger server and store the policies locally using a REST API. When a user request comes to any Hadoop component, the plugin intercepts the request and evaluates it against the security policy to decide if the user request should be authorized or not.

The advantages of using Apache Ranger with Hadoop are primarily associated with centralizing the security administration of many security settings and tasks through in a central WebUI. Although there are some areas important to cluster security, such as OS security settings, that are not managed by Ranger. It helps to provide consistent AC across all Hadoop components with Ranger plugins. Use of the User Sync and TagSync features are instrumental to fine grained AC, facilitating ABAC style security.

Apache Atlas

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

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

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

When we installed Apache Atlas, we selected H-Base for the metadata store and Solar for the index store, which is the default configuration. Other core components are the graph engine that supports relationships between entities. For example, columns and database, files and folder relationships are managed by graph engine. The ingest/export features captures messages from the metadata sources, creates entities, and updates events.

A challenge we observed with Apache Atlas is that nothing is done automatically by Atlas. You have to launch the hooks, i.e., curl commands in scripts that run regularly on the component systems to crawl through your data store. Although some hooks are available from the Apache Atlas repository for installation and configuration on the core components. For other data sources, you will have to write hooks, using the curl calls for registering and tagging the entities.

Analysis and Observations

The challenge with a distributed data processing framework is that the security issues previously contained within a computer system are now across a network of computers with applications and data in multiple locations. In addition to the observations of the component AC security services previously described, we observed several issues in the system-wide approach to security. We also captured performance information under different security configurations. Apache Hadoop provides impressive performance and further security enhancements will continue to increase its popularity. An enhanced system security strategy that includes security testing will enable its application to multi-tenant, multi-sensitivity level data analysis challenges.

ABAC Support

A primary goal for our experimental configuration was to examine the ability to support fine grained ABAC. Our experiments in this area indicate that the framework is in place to make this

feasible, however this requires significant software configurations, including writing scripts to connect the metadata into the policy enforcement program. There is a high risk of misconfiguration.

ABAC can be achieved using Apache Ranger policies that bring together attributes from three sources, as depicted in Figure 21. Using the Apache Ranger User Sync, user information such as group assignments can be provided from the directory service over LDAP. Ranger also provide plugins to HDFS and YARN, so file permissions and job scheduling can be controlled. The Tag Sync capability with Atlas allows the HDFS and YARN metadata to be used as a basis of security policies. Due to the challenges associated with using open-source software, the scope of the experiment was limited. However, this same strategy can be used with Apache Hive and other components. Based upon our analysis we believe the classification tags from Atlas and policies created in Ranger provide capabilities to largely achieve ABAC, however, there are limitations associated with flexibility in assigning attributes to users. Users can be managed locally in Ranger or externally through a directory service and assigned to groups, which could be mapped to the concept of attributes for policy decisions. Assigning users to multiple groups and then using those groups as part of the policy decision process provides a concept of attribute based control, however, this may not provide the flexibility of key-value pair attribute based controls.

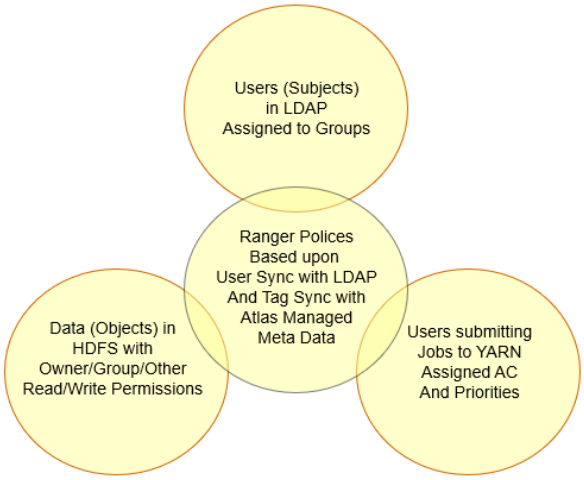


Figure 22: ABAC Implementation using HDFS, LDAP, YARN, Apache Ranger and Atlas Multiple Management Consoles

Although there are several management frameworks for Apache Hadoop, such as Apache Ambari and Cloudera Manager, a single management console that fully integrates across the ecosystem are not available. System security management interfaces, such as the Apache Ranger Admin WebUI, is 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 they manage. You 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 to configure or “turn-on” security components, however they do not confirm the system is fully secured. Validating security policy enforcement requires interfacing to separate consoles, including directory systems and the Ranger Admin WebUI. Confirmation that the system is correctly configured and reviewing security status information requires examining multiple interfaces and files. Automating collection of this status information would require significant development of custom scripts that push logs to a central location.

Verification of Security Software

Software that provides security functionality should be subject to rigorous, 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. Government and businesses prefer the RedHat version of Linux because of its independent certification [132].

The approach of independently writing security software and hooks to interface to critical security services, such as the Kerberos Key Distribution Center (KDC) 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 [133].

Veracode's State of Software Security Report found that more than three-quarters (75.2 percent) of applications have security flaws [134]. This is among large software manufacturers and does not necessarily include small rapidly developed glue code developed under business pressures to get BDP investments up and running.

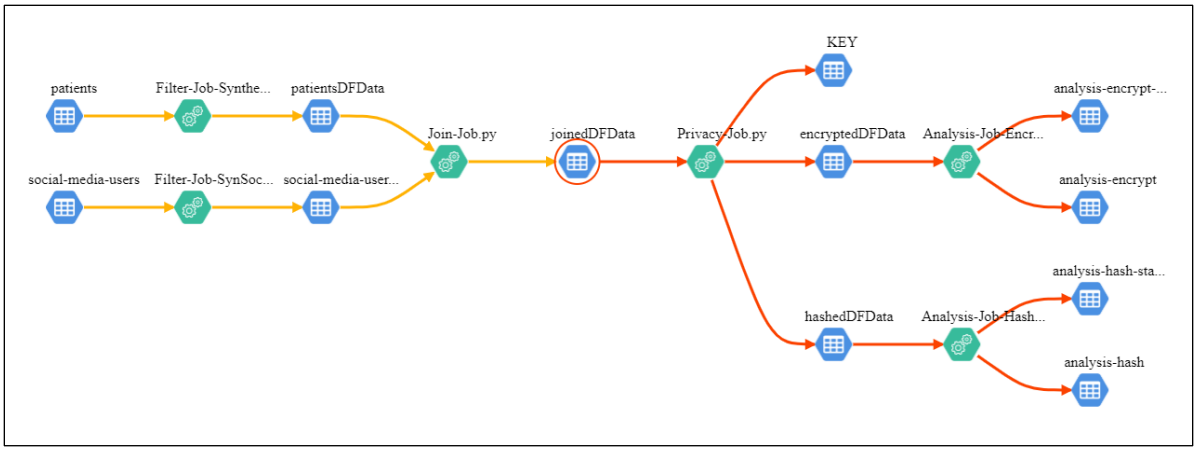
In conducting this analysis, we noted several security issues in software components. Many of the configuration files contain administrator passwords in clear text. Log files, which can contain sensitive information such as Kerberos service tickets, are stored in multiple, potentially unprotected locations. The data encryption algorithms used in Apache Hadoop are 3DES and R4, which are relatively weak [135]. Additional configuration is required to incorporate AES. Rigorous in-depth security testing would identify these types of design flaws for remediation.

Performance

We analyzed the performance of Apache Hadoop under different security configurations. Specifically, we measured the elapsed execution time for each sequential job. The workflow of

data inputs, PySpark programs, and outputs was assigned attributes and tags using Apache Atlas, as shown in the Figure 23.

Figure 23: Data Processing and Attribute Classification Propagation in Apache Atlas



For our experiment, the elapsed execution time was gathered from the Hadoop Resource Manager Web GUI in the different security configurations, as listed in Table 14 and shown in Figure 24. The various security configurations are: execution with the Ranger plugins disabled, execution as the Hadoop local privileged user, execution with resource based AC using Apache Ranger policies, and execution with tag-based AC which used both Apache Ranger polices and Apache Atlas metadata. The fourth configuration corresponds to an ABAC approach. The current performance analysis did not indicate any significant differences in executing the sequence of data processing programs (PySpark programs) in the data lifecycle use case. Overall, we noted very similar program execution times in all three configurations. This indicates the AC security services had very limited impact to the performance of the system.

Figure 24: Elapsed Time Performance Analysis Summary

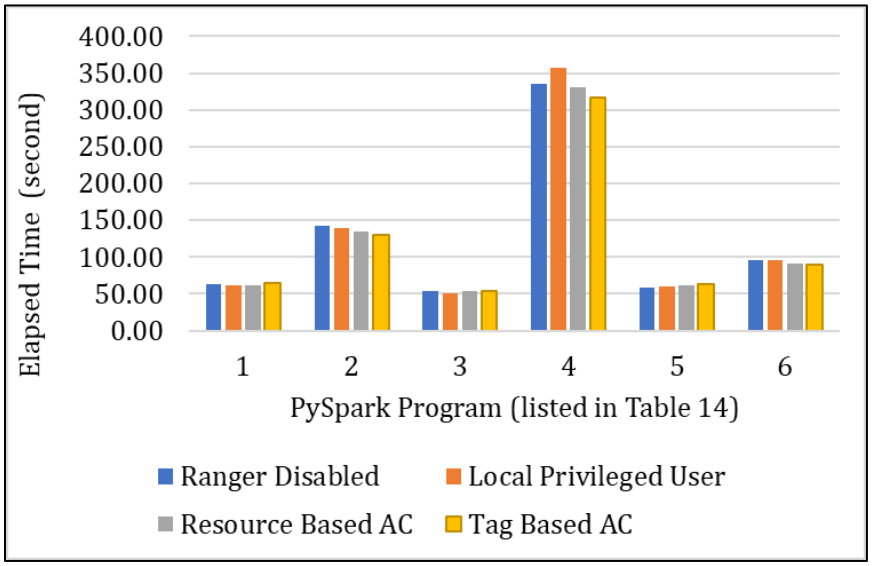


Table 14: AC Security Impact on Program Execution Time

PySpark Analytic Program	Description	Elapsed Times (seconds), (average of 15 trials)			
		Ranger Plugins Disabled	Local Privileged User	Resource Based AC	Tag Based AC
1 - Filter-Job-SynSocial.py	Filters Twitter JSON data from the social media data set and puts it into Spark DataFrames	62.50	61.20	61.93	64.00
2 - Filter-Job-Synthea.py	Filters HL7 JSON data from the healthcare data set and puts it into Spark DataFrames	142.73	139.13	133.93	129.67
3 - Join-Job.py	Joins the social media and healthcare data sets based upon the association of the social media message to the 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 the hashed data from the privacy job to produce summary message counts per medical condition	58.00	60.33	61.07	62.73
6 - Analysis-Job-Encrypt.py	Analyzes the encrypted data from the privacy job to produce summary message counts per medical condition	95.20	95.20	91.67	89.40

Although the expectation is that there would be at least a minor, detectable performance difference in executing the data lifecycle in different security configurations, based upon industry reported performance analysis, we would need to scale out significantly to see minor impacts on performance [136]. Our data set represents 1,000 users/patients, is 14 GB in size and configured with block replication set to 3. We estimate we would need to scale up from 14 GB to over 1 TB of data, which would require 6 or more data nodes for a minor (under 10%) performance impact.

As the size of the data set and processing lifecycle complexity are scaled out, performance impact may be identified by increase in the elapsed time. However, the overhead added by the AC services provided by Apache Ranger is expected to be minimal. Encryption of data and Kerberos add a more significant increase to the elapsed time.

The Transaction Processing Performance Council (TPC) has several benchmarks for Hadoop ecosystem big data set analysis. The TPC Benchmark™HS (TPCx-HS) benchmark workload is based upon the Hadoop TeraSort program, for example [137]. These benchmarks provide insights for acquiring hardware for a Hadoop cluster, but do not address security specific performance impacts, such as encryption.

Additional tools for monitoring performance of a large-scale Hadoop cluster are available. The AWS EC2 interface also provides statistics on 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 is available for use, such as Ganglia and Nagios. 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 graphical user interface (GUI). Hadoop specific tools such as Apache Ambari and Cloudera manager must be

used to install all the Hadoop ecosystems components under management and cannot be added to an installed cluster. These tools provide Hadoop specific statistics, such as data block replication status and YARN job schedule management.

CHAPTER SIX: CONCLUSION

Overall, if Apache Hadoop is used in business applications, a commercial, managed distribution is required. This must be used to provide the add-ons and integration glue code necessary for logging, security services, and management center command and control. A professional, team approach is needed to handle the increasingly complex environment when larger more diverse tools are used to meet analysis requirements. 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 in each component.

However, for researchers, using open-source Apache Hadoop ecosystem components provides a great opportunity to analyze the security features in detail. Commercial versions can hide security features in compiled code. Detailed analysis of potential vulnerabilities supports rigorous testing and new innovative approaches to security.

Recommendations

Three areas, in particular, should be advanced and incorporated into the design 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, and

(3) A complete central management system that provides a single console to status all the system components.

Without these fundamental controls, more advanced security services, such as ABAC will be difficult to achieve in a complete and integrated manner.

Secure Data Analyst Notebook

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

Preference should be made on 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. Jupyter notebooks for example, depend upon a gateway-enabled notebook sever 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. When users can take on the superuser credentials unintentional damage could occur.

Data Security Service Layer

The Hadoop ecosystem is based upon a group of independently developed projects that are connected using open protocols rather than being built from the ground up with an integrating security service layer. This results in access with accounts with root-level privileges being widely used to overcome software problems. A significant challenge in setting up a BDP ecosystem is that all the software components have different version update schedules and not all versions are compatible with each other. Enforcing a common security service layer that needs to

be invoked to interoperate would help to ensure security services are not disabled or bypassed as software versions change.

The status of Apache Hadoop security features is that they are added on, rather than being 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 [138] for data processing systems and the Generic Security Standard Application Programing Interface (GSS-API) for software development [139].

Apache Ranger is a correct step in this direction, however the developers' goal to be flexible and extensible, can undermine consistency in component security services. Each capability can be optionally implemented which can result in fractured security. Apache Ranger needs to evolve to a complete, integrated central security service that applies standard interfaces, such as XACML. This integrated, comprehensive security service would provide a back plain that is invoked consistently by all BDP system services.

Central Management View

Given the importance of BDP, system security managers and administrators need to think and operate like a cybersecurity attacker. A central management view is needed that supports this philosophy and provides complete security status information. This would be the result of active, continuous testing of each ecosystem component and the collective environment. This active testing needs to occur across the cybersecurity lifecycle. Penetration scanning that attempts to exploit open ports, protocols, and services to gain a foothold could be provided using tools such as Nmap and OpenVAS. The exposure to exploits that exfiltrate or corrupt data by establishing

command and control through campaign kits such as Metasploit need to be well understood. Any SQL input should be subject to SQL injection testing in a manner that informs the security team.

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

Traffic analysis 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. Overall, the least privilege principle needs to be applied by ensuring individual accountability and restricting shared use of superuser accounts.

Application of Findings

The methodology we applied to conduct this research focused on the AC, specifically ABAC area of security for BDP, using the Apache Hadoop ecosystem as the model framework. Our design of experiment can be used to research other areas, such as the capacity scheduler in the YARN Resource Manger to further ensure effective shared use and prevent unauthorized jobs, such as crypto-currency mining. Other security research areas that our experiment framework could be applied to is evaluating the potential to bypass encryption and threats from inference in any unencrypted data such as logs or temporary files. The ability to look at all the back-end configuration files and settings in open source software provides great insights when considering emerging trends in security, such as zero-trust architectures and gaining insights from advanced machine learning and artificial intelligence algorithms.

Summary

With the availability next-generation BDP tools, such as Apache Hadoop and cloud-based Hadoop-like ecosystems, organization are seeing the potential to store and process larger volumes and different types of data. This includes sensitive data that needs to be protected, not only to meet compliance regulations, but also to protect the investments in high integrity data that informs business decisions. Our analysis of Apache Hadoop security, focusing on ABAC, identified an approach to analyze vulnerabilities and potential attacks. Based upon our experimental environment, we have identified several security issues. Also, we have made our results available to support additional, related research. An overall security goal for BDP is to achieve fine grain AC through ABAC. In the process of conducting this analysis we identified three main areas that we recommend for further research and experimentation to reach this goal. That is 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 enhancements need to be developed through continued BDP security research.

LIST OF REFERENCES

[1] A. M. Tall, J. Wang and D. Han, "Survey of data intensive computing technologies application to security log data management," in *3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*, Shanghai China, 2016.

[2] A. M. Tall, C. C. Zou and J. Wang, "Integrating Cybersecurity Into a Big Data Ecosystem," in *IEEE MILCOM*, San Diego, CA, 2021.

[3] NIST Public Big Data Working Group, "Big Data Interoperability Framework," NIST, U.S. Dept. of Commerce, Gaithersburg, MD, 2019.

[4] NIST Big Data Public Working Group, "NIST Big Data Ineroperability Framework: Volume 4, Security and Privacy, Version 3, SP 1500-4r2," U.S. Dept. of Commerce, Gaitersburg, MD, 2019.

[5] Research Data Alliance (RDA), "Big Data Interest Group (IG)," [Online]. Available: https://www.rd-alliance.org/groups/big-data-analytics-ig.html.

[6] National Security Agency (NSA) Central Security Service (CSS), "Defense in Depth," National Security Agency, Fort Meade, MD, 2010.

[7] J. Ronan and contributors, "The Hadoop Ecosystem Table," GitHub Pages, 2021. [Online]. Available: https://hadoopecosystemtable.github.io/.

[8] T. White, Hadoop: The Definitive Guide, 4th ed., Sebastopol, CA: O'Reilly Media, 2015.

[9] BDBSRAWG - Guide for Big Data Business Security Risk Assessment , "IEEE 2813-2020 - IEEE Standard for Big Data Business Security Risk Assessment," IEEE CTS/SC Standards Committee, New York, 2021.

[10] NIST Information Technology, "Measurements for Information Security," U.S. Dept. of Commerce, 15 Sep. 2020. [Online]. Available: https://www.nist.gov/cybersecurity/measurements-information-security.

[11] Y. Cheng, J. Deng, J. Li, S. DeLoach, A. Singhal and X. Ou, "Metrics of Security," NIST, Gaithersburg, MD, 2014.

[12] D. Flater, "Bad Security Metrics Part 2: Solutions," *IEEE IT Professional,* vol. 20, no. 2, pp. 76-79, 2018.

[13] Q. Liu, L. Xing and C. Zhou, "Probabilistic modeling and anlaysis of sequential cyber-attacks," *Wiley Engineering Reports,* vol. 1, no. 4, 2019.

[14] S. Chen, Z. Kalbarczyk, J. Xu and R. K. Iyer, "A data-driven finite state machine model for analyzing security vulnerabilities," in *International Conference on Dependable Systems and Networks*, San Francisco, CA, 2003.

[15] F. Dang, H. Liang, S. Li, D. Li and H. Liu, "Design and Implementation of Computer Network Information Security Protection Based on Secure Big Data," in *IEEE 3rd IICSPI*, Chongquing City, China, 2020.

[16] F. Wang, H. Wang and L. Xue, "Research on Data Security in Big Data Cloud Computing Environment," in *IEEE 5th IAEAC*, Chongqing, China, 2021.

[17] X. Sun, P. Liu and A. Singhal, "Toward Cyberresiliency in the Context of Cloud Computing," *IEEE Security & Privacy,* vol. 16, no. 6, pp. 71-75, 2018.

[18] B. Spivey and J. Echeverria, Hadoop Security Protecting Your Big Data Platform, Sebastopol, CA: O'Reilly Media, 2015.

[19] National Institute of Standards and Technology (NIST), "Risk Management Framework SP 800-37 Rev. 2," NIST, U.S. Dept. of Commerce, Gaithersburg, MD, 2021.

[20] P. J. Velthuis, "New authentication mechanism using certificates for big data analytic tools," KTH Royal Inst. of Tech., Stockholm, SE, 2017.

[21] S. Sinha, S. Gupta and A. Kumar, "Emerging Data Security Solutions in Hadoop based Systems: Vulnerabilities and Their Counermeasures," in *IEEE ICCCIS*, Greater Noida, India, 2019.

[22] J. Longstaff and J. Noble, "Attribute based access control for big data applications by query modification," in *IEEE Second International Conference on Big Data Computing Service and Applications*, Oxford, UK, 2016.

[23] S. Oulmakhzoune, N. Cuppens-Boulahia, F. Cuppens, S. Morucci, M. Barhamgi and D. Benslimane, "Privacy query rewriting algorithm instrumented by a privacy-aware access control model," *Annals of Telecommunications,* vol. 69, no. 1-2, pp. 3-19, 2014.

[24] K. Zhang, X. Zhou, Y. Chen, X. Wang and Y. Ruan, "Sedic: privacy-aware data intensive computing on hybrid clouds," in *ACM Conf. on Computer and Comm. Security*, Chicago, IL, 2011.

[25] N. Khamphakdee, N. Benjamas and S. Saiyod, "Performance Evaluation of Big Data Technology on Designing Big Netowrk Traffic Data Analysis System," in *ISCIS*, Sapporo, Japan, 2016.

[26] G. S. Bhathal and A. Singh, "Big Data: Hadoop framework vulnerabilities, security issues and attacks," *Elsevier Array,* Vols. 1-2, p. 100002, 2019.

[27] J. Wang, D. Crawl, S. Purawat, M. Nguyen and I. Altintas, "Big data provenance: Challenges, state of the art and opportunities," in *IEEE International Conf. on Big Data*, Santa Clara, CA, 2015.

[28] NIST, ITL, CSD, "Standards for Security Categorization of Federal Information and Informatoin Systems, FIPS PUB 199," NIST, U.S. Dept. of Commerce, Gaithersburg, MD, 2004.

[29] N. O. Leslie, R. E. Harang, L. P. Knachel and A. Kott, "Statistical Models for the Number of Successful Cyber Intrusions," *Journal of Defense Modeling and Simulation,* vol. 15, no. 1, pp. 49-63, 2018.

[30] Varonis, "Cybersecurity: the motivation behind cyber-hacks," Big Data Made Simple, 30 Jul. 2019. [Online]. Available: https://bigdata-madesimple.com/cybersecurity-the-motivation-behind-cyber-hacks-infographic/.

[31] AT&T Business - Cybersecurity, "Understanding cyber attacker motivations to best apply controls," AT&T, 19 Feb. 2020. [Online]. Available: https://cybersecurity.att.com/blogs/security-essentials/understanding-cyber-attacker-motivations-to-best-apply-controls.

[32] PurpleSec, "2021 Cyber Security Statistics, The Ultimate List of Stats, Data & Trends," PurpleSec LLC, 29 Apr 2021. [Online]. Available: https://purplesec.us/resources/cyber-security-statistics/.

[33] Verizon, "Data Breach Investigations Reports (DBIR)," Verizon, New York, 2021.

[34] DASD, DT&E, "The Department of Defense Cyber Table Top Guidebook," DoD, Washington, D.C., 2018.

[35] MITRE-Engenuity, "Security Control Mappings: A Bridge to Threat-Informed Defense," The MITRE Corp., 15 Dec. 2020. [Online]. Available: https://ctid.mitre-engenuity.org/our-work/nist-800-53-control-mappings/.

[36] Center for Threat Informed Defense, "attack-control-framework-mappings," GitHub, Dec. 2020. [Online]. Available: https://github.com/center-for-threat-informed-defense/attack-control-framework-mappings.

[37] CCF, "Common Criteria for Information Technology and Security Evaluation, ISO 15408," ISO/IEC, virtual, 2017.

[38] Office of the Under Secretary of Defense, "Cybersecurity Maturity Model Certification," DoD, Washington, D.C., 2020.

[39] W. S. Humphrey, "Characterizing the software process: a maturity framework," *IEEE Software,* vol. 5, no. 2, pp. 73-79, March 1988.

[40] Forbes and IBM, "Forbes Insights Fallout The Reputational Impact of IT Risk," Forbes, Jersey CIty, NJ, 2014.

[41] A. M. Tall, C. C. Zou and J. Wang, "Access Control in the Era of Big-Data Driven Models and Simulations," in *I/ITSEC*, Orlando, FL, 2019.

[42] E. Bertino, G. Ghinita and A. Kamra, "Access Control for Databases: Concepts and Systems," *Foundations and Trends® in Databases,* vol. 3, no. 1-2, pp. 1-148, 2011.

[43] M. A. Harrison, W. L. Ruzzo and J. D. Ullman, "Protection in operating systems," *Communications of the ACM,* vol. 19, no. 8, pp. 461-471, August 1976.

[44] V. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller and K. Scarfone, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations SP 800-162," National Institute of Standards and Technology (NIST), Gaithersburg, MD, 2014.

[45] D. Ferraiolo, R. Chandramouli, V. Hu and R. Kuhn, "A Comparison of Attribute Based Access Control (ABAC) Standards for Data Service Applications Extensible Access Control Markup Language (XACML) and Next Generation Access Control (NGAC) SP 800-178," National Institute for Standards and Technology (NIST), Gaithersburg, MD, 2016.

[46] InterNational Committee for Information Technology Standards (INCITS), *Information technology - Next Generation Access Control - Implementation Requirements, Protocols and API Definitions (NGAC-IRPAD),* 0.75 ed., vol. 525:201x, C. T. Committee, Ed., Washington, DC: InterNational Committee for Information Technology Standards, 2019, pp. 1-44.

[47] J. Walonoski, M. Kramer, J. Nichols, A. Quian, C. Moesel, D. Hall, C. Duffett, K. Dube, T. Gallagher and S. McLachlan, "Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record," *Journal of the American Medical Informatics Association (JAMIA),* vol. 25, no. 3, pp. 230-238, 30 March 2018.

[48] O. Marchesini, "Advanced Analytics for EXtremely Large European Databases," Porta Vita Networked Health, - - -. [Online]. Available: https://www.portavita.com/axle. [Accessed 1 March 2022].

[49] B. Fisher, N. Brickman, P. Burden, S. Jha, B. Johnson, A. Keller, T. Kolovos, S. Umarji and S. Weeks, "Attribute Based Access Control, Volume B: Approach, Architecture and Security Characteristics SP 1800-3B," National Institute of Standards and Technology (NIST), Gaithersburg, MD, 2017.

[50] V. Hu, D. Ferraiolo, R. Kunn, A. Schnitzer, K. Sandlin, R. Miller and K. Scarfone, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations SP 800-162," NIST, Gaithersburg, MD, 2014.

[51] S. Sen, S. Guha, A. Datta, S. Rajamani, J. Tsai and J. Wing, "Bootstrapping privacy compliance in big data systems," in *2014 IEEE Symposium on Security and Privacy*, San Jose, CA, 2014.

[52] Z. Zhioua and R. Ameur-Boulifa, "Framework for the Formal Specification and Verification of Security Guidelines," *Advances in Science, Technology and Engineering Systems (ASTES) Journal,* vol. 3, no. 1, pp. 38-48, 30 January 2018.

[53] B. Bezawada, K. Haefner and I. Ray, "Securing Home IoT Environments with Attribute-Based Access Control," in *ABAC'18: Proceedings of the Third ACM Workshop on Attribute-Based Access Control*, Tempe AZ, 2018.

[54] V. Hu, D. Ferraiolo and R. Kuhn, "Attribute Considerations for Access Control Systems," National Institute of Standards and Technology (NIST), Giathersburg, MD, 2019.

[55] D. Nguyen, *Provenance-Based Access Control Models Phd thesis,* San Antonio TX: University of Texas at San Antonio, Department of Computer Science, 2014.

[56] C. Liao and A. Squicciarini, "Towards Provenance-Based Anomaly Detection in MapReduce," in *IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID)*, Shenzhen China, 2015.

[57] L. Sun, J. Park, D. Nguyen and R. Sandhu, "A Provenance-Aware Access Control Framework with Typed Provenance," *IEEE Transactions on Dependable and Secure Computing,* vol. 13, no. 4, pp. 411-423, 2016.

[58] H. Won, M. C. Nguyen, M.-S. Gil and Y.-S. Moon, "Advanced Resource Management with Access Control for Multitenant Hadoop," *Journal of Communications and Networks,* vol. 17, no. 6, pp. 592-601, 2015.

[59] N. Solanki, Y. Huang, I.-L. Yen, F. Bastani and Y. Zhan, "Resource and Role Hierarchy Based Access Control for Resourceful Systems," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Tokyo, Japan, 2018.

[60] Y. Yu, Y. Chen and Y. Wen, "Task-role based access control model in logistics management system," in *Proceedings of 2013 IEEE International Conference on Service Operations and Logistics, and Informatics*, Dongguan, China, 2013.

[61] S. Alshammari, A. Albeshri and K. Alsubhi, "Integrating a High-Reliability Multicriteria Trust Evaluation Model with Task Role-Based Access Control for Cloud Services," *Symmetry,* vol. 13, no. 3, p. 492, 2021.

[62] P. Wang and L. Jiang, "Task-role-based Access Control Model in Smart Health-care System," in *MATEC Web of Conferences International Conference on Engineering Technology and Application (ICETA 2015)*, Nagoya, Japan, 2015.

[63] L. Ma, L. Tao, Y. Zhong and K. Gai, "RuleSN: Research and Application of Social Network Access Control Model," in *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, 2016.

[64] Y. Cheng, J. Park and R. Sandhu, "An Access Control Model for Online Social Networks Using User-to-User Relationships," *IEEE Transactions on Dependable and Secure Computing,* vol. 13, no. 4, pp. 424-436, 2016.

[65] S. Z. R. Rizvi, P. W. Fong, J. Crampton and J. Sellwood, "Relationship-Based Access Control for an Open-Source Medical Records System," in *SACMAT '15: Proceedings of the 20th ACM Symposium on Access Control Models and Technologies*, Vienna Austria, 2015.

[66] L. Ma, L. Tao, K. Gai and Y. Zhong, "A novel social network access control model using logical authorization language in cloud computing," *Concurrency and Computation Practice and Experience,* vol. 29, no. 14, pp. 1-17, 2016.

[67] R. Zhang, L. Liu and R. Xue, "Role-based and time-bound access and management of EHR data," *Security and Communication Networks,* vol. 7, no. 6, pp. 994-1015, 2014.

[68] K. Yang, Z. Liu, X. Jia and X. S. Shen, "Time-Domain Attribute-Based Access Control for Cloud-Based Video Content Sharing: A Cryptographic Approach," *IEEE Transactions on Multimedia,* vol. 18, no. 5, pp. 940-950, 2016.

[69] M. Gupta, F. Patwa and R. Sandhu, "Object-Tagged RBAC Model for the Hadoop Ecosystem," in *IFIP Annual Conference on Data and Applications Security and Privacy DBSEC 2017*, Philadelphia, PA, 2017.

[70] A. Kayes, J. Han and A. Colman, "An ontological framework for situation-aware access control of software services," *Information Systems,* vol. 53, no. C, pp. 253-277, 2015.

[71] A. Kumar TK, H. Liu, J. P. Thomas and X. Hou, "Content sensitivity based access control framework for Hadoop," *Digital Communications and Networks,* vol. 3, no. 4, pp. 213-225, 2017.

[72] W. Zeng, Y. Yang and B. Luo, "Access control for big data using data content," in *2013 IEEE International Conference on Big Data*, Silicon Valley, CA, 2013.

[73] R. ". Morgan, S. Cantor, S. Carmody, W. Hoehn and K. Klingenstein, "Federated Security: The Shibboleth Approach," *EDUCASE Quarterly,* pp. 12-17, 1 January 2004.

[74] OASIS, *Cross-Enterprise Security and Privacy Authorization (XSPA) Profile of SAML v2.0 for Healthcare, Version 2.0, Committee Specification 01,* Burlington, MA: OASIS, 2019.

[75] HL7 International, *HL7 Healthcare Privacy and Security Classification System (HCS), Release 1,* Ann Arbor, MI : HL7 International, 2014.

[76] X. Fu, X. Nie, T. Wu and F. Li, "Large universe attribute based access control with efficient decryption in cloud storage system," *Journal of Systems and Software,* vol. 135, no. -, pp. 157-164, 2018.

[77] J. Li, Y. Zhang, J. Ning, X. Huang, G. S. Poh and D. Wang, "Attribute Based Encryption with Privacy Protection and Accountability for CloudIoT," *IEEE Transactions on Cloud Computing (Early Access),* Vols. -, no. -, p. 1, 2020.

[78] W. Teng, G. Yang, Y. Xiang, T. Zhang and D. Wang, "Attribute-Based Access Control with Constant-Size Ciphertext in Cloud Computing," *IEEE Transactions on Cloud Computing,* vol. 5, no. 4, pp. 617-627, 2017.

[79] J. Wang, D. Crawl, S. Purawat, M. Nguyen and I. Altintas, "Big data provenance: Challenges, state of the art and opportunities," in *2015 IEEE International Conference on Big Data*, Santa Clara, CA, 2015.

[80] J. Hellerstein, V. Sreekanti, J. Gonzalez, J. Dalton, A. Dey, S. Nag, K. Ramachandran, S. Arora, A. Bhattacharyya, S. Das, M. Donsky, G. Fierro, C. She, C. Steinbach, V. Subramanian and E. Sun, "Ground: A Data Context Service," in *CIDR 2017*, Chaminade, CA, 2017.

[81] Y. Sowmy, M. Nagaratna and C. Shoba Bindu, "M-SANIT: a Framework for Effective Big Data," *Journal of Theoretical and Applied Information Technology,* vol. 96, no. 6, pp. 1596-1605, 2018.

[82] S. Nagajothi and N. Raj Kumar, "Data Anonymization Technique for Privacy Preservation Using MapReduce Framework," *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 5, no. 5, pp. 1012-1018, 2016.

[83] X. Zhang, L. Yang, C. Liu and J. Chen, "A Scalable Two-Phase Top-Down Specialization Approach for Data Anonymization Using MapReduce on Cloud," *IEEE Transactions on Parallel and Distributed Systems,* vol. 25, no. 2, pp. 363-373, 2014.

[84] B. Saraladevi, N. Pazhaniraja, P. Victer Paul, M. Saleem Basha and P. Dhavachelvan, "Big Data and Hadoop-a Study in Security Perspective," *Procedia Computer Science,* vol. 50, no. -, pp. 596-601, 2015.

[85] C. T. Hu, D. Kuhn and D. Yaga, "Verification and Test Methods for Access Control Policies/Models SP 800-192," National Institute of Standards and Technologies (NIST), Gaithersburg, MD, 2017.

[86] Cloud Security Aliance, Top Threats Working Group, "Top threats to Cloud COmputing: Egregious Eleven," Cloud Security Aliance, Seattle, WA, 2019.

[87] S. Khandelwal, *Insecure hadoop clusters expose over 5,000 terabytes of data,* The Hacker News, 2017.

[88] G. S. Bhathal and A. Singh, "Big Data: Hadoop framework vulnerabilities, security issues and attacks," *Array,* Vols. 1-2, no. 100002, pp. 1-8, 2019.

[89] S. Jha, S. Sural, V. Atluri and J. Vaidysa, "Security Analysis of ABAC under an Administrative Model," *IET Information Security,* vol. 13, no. 2, pp. 96-103, 2019.

[90] E. Mouzakitis, *How to monitor hadoop metrics,* -: DataDog, 2016.

[91] D. Gros, M. Blanc, J. Briffaut and C. Toinard, "PIGA-cluster: A distributed architecture integrating a shared and resilient reference monitor to enforce mandatory access control in the HPC environment," in *International Conference on High Performance Computing & Simulation (HPCS)*, Helsinki, Finland, 2013.

[92] Z. Dou, I. Khalil, A. Khreishah and A. Al-Fuqaha, "Robust Insider Attacks Countermeasure for Hadoop: Design and Implementation," *IEEE Systems Journal,* vol. 12, no. 2, pp. 1874-1885, 2018.

[93] J. Cohen and S. Acharya, "Incorporating hardware trust mechanisms in Apache Hadoop: To improve the integrity and confidentiality of data in a distributed Apache Hadoop file system: An information technology infrastructure and software approach," in *2012 IEEE Globecom Workshops*, Anaheim, CA, 2012.

[94] B. Spivey and J. Echeverria, Hadoop Security, page 48, Sebastopol, CA: O'Reilly Media, 2015, p. 48.

[95] T. Nguyen, M. Gondree, J. Khosalim and C. Irvine, "Towards a Cross-Domain MapReduce Framework," in *IEEE Military Communications Conference*, San Diego, CA, 2013.

[96] A. M. Tall, C. C. Zou and J. Wang, "Generating Connected Synthetic Electronic Health Records and Social Media Data for Modeling and Simulation," in *I/ITSEC*, Orlando, 2020.

[97] J. Brown, "Using Social Media Data to Identify Outbreaks and Control Disease," *Government Technology,* p. 1, 7 January 2015.

[98] S. Jordan, S. Hovet, I. C.-H. Fung, H. Liang, K.-W. Fu and Z. T. H. Tse, "Using Twitter for Public Health Surveillance from Monitoring and Prediction to Public Response," *Data Special Issue Big Data and Digital Health,* vol. 4, no. 1, p. 6, 2019.

[99] T. Kahara, K. Haataja and P. Toivanen, "A novel recommendation system approach utilizing social network profiles," in *13th International Conference on Hybrid Intelligent Systems (HIS 2013)*, Gammarth, Tunisia, 2013.

[100] J. Stromberg, "Your Tweets Can Predict When You'll Get the Flu," *Smithsonian Magazine,* p. 1, 8 November 2013.

[101] Geeta and R. Niyogi, "Demographic analysis of Twitter users," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Jaipur, India, 2016.

[102] K. Singh, S. Dhawan and Pratibha, "Real-time data elicitation from Twitter: Evaluation and depiction strategies of tweets concerned to the blazing issues through Twitter application," in *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*, Noida, India, 2014.

[103] A. Sadilek, H. Kautz and V. Silenzio, "Predicting disease transmission from geo-tagged micro-blog data," in *AAAI'12: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, Toronto Ontario Canada, 2012.

[104] E. Bucher, C. Fieseler and M. Meckel, "Beyond Demographics -- Explaining Diversity in Organizational Social Media Usage," in *2013 46th Hawaii International Conference on System Sciences*, Wailea, HI, 2013.

[105] A. Schwartz, "Twitter Knows When You'll Get Sick Before You Do," *FastCompany,* p. 1, 12 August 2012.

[106] P. Sneiderman, *Using Twitter to Track the Flu: Researchers Find a Better Way to Screen the Tweets,* Baltimore, MD: Johns Hopkins University, 2013.

[107] A. Smith and M. Anderson, *Social Media Use in 2018,* Washington D.C.: Pew Research Center, 2018.

[108] Y. Sagduyu, A. Grushin and Y. Shi, "Synthetic Social Media Data Generation," *IEEE Transactions on Computational Social Systems,* vol. 5, no. 3, pp. 605-620, 2018.

[109] C. Yu, F. Xia, Q. Zhang, H. Ma, W. Qian, M. Zhou, C. Jin and A. Zhou, "BSMA-Gen: A Parallel Synthetic Data Generator for Social Media Timeline Structures," in *International Conference on Database Systems for Advanced Applications DASFAA 2014: Database Systems for Advanced Applications*, Bali, Indonesia.

[110] T. H. Moreira de Oliveira and M. Painho, "Emotion & stress mapping: Assembling an ambient geographic information-based methodology in order to understand smart cities," in *2015 10th Iberian Conference on Information Systems and Technologies (CISTI)*, Aveiro, Portugal, 2015.

[111] A. H. Yazdavar, M. S. Mahdavinejad, G. Bajaj, K. Thirunarayan, J. Pathak and A. Sheth, "Mental Health Analysis Via Social Media Data," in *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, New York, NY, 2018.

[112] Q. C. Nguyen, H. Meng, D. Li, S. Kath, M. McCullough, D. Paul, P. Kanokvimankul, T. X. Nguyen and F. Li, "Social media indicators of the food environment and state health outcomes," *Public Health,* vol. 148, no. Epub, pp. 120-128, 2017.

[113] F. Xia, Y. Li, C. Yu, H. Ma and W. Oian, "BSMA: a benchmark for analytical queries over social media data," *Proceedings of the VLDB Endowment,* vol. 7, no. 13, pp. 1573-1576, 2014.

[114] S. Rosenthal, S. Mohammad, P. Nakov, A. Ritter, S. Kiritchenko and V. Stoyanov, "SemEval-2015 Task 10: Sentiment Analysis in Twitter," *SemEval-2015,* vol. arXiv preprint arXiv:1912.00741, no. -, p. 14, 2019.

[115] I. Chandrakar and V. R. Hulipalled, "Privacy Preserving Big Data mining using Pseudonymization and Homomorphic Encryption," in *2021 2nd Global Conference for Advancement in Technology (GCAT)*, Bangalore, India, 2021.

[116] R. Sellami, F. Zalila, A. Nuttinck, S. Dupont, J.-C. Deprez and S. Mouton, "FADI - A Deployment Framework for Big Data Management and Analytics," in *2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, Bayonne, France, 2020.

[117] P. Colombo and E. Ferrari, "Access Control in the Era of Big Data: State of the Art and Research Directions," in *23nd ACM on Symposium on Access Control Models and Technologies (SACMAT '18)*, Indianapolis Indiana USA, 2018.

[118] N. Horton and A. DeSimone, "Sony's Nightmare before Christmas: The 2014 North Korean Cyber Attack on Sony and Lessons for US Government Actions in Cyberspace," Defense Technical Information Cenger, Laurel Maryland USA, 2018.

[119] H. Saleem and M. Naveed, "SoK: Anatomy of data breaches," *Proceedings on Privacy Enhancing Technologies,* no. 4, pp. 153-174, 1 October 2020.

[120] M. Hart, *Kerberos Attacks: What You Need to Know,* Newton, MA: Cyberark, 2015.

[121] L. George, *User name handling in Hadoop,* Wedel, Hamburg: OpenCore, 2016.

[122] ASF Infrabot, "Powered By Apache Hadoop," 18 December 2020. [Online]. Available: http://wiki.apache.org/hadoop/PoweredBy.

[123] G. S. Bhathal and A. Singh, "Big Data: Hadoop framework vulnerabilities, security issues and attacks," *Array,* pp. 1-8, January-April 2019.

[124] X. Fu, Y. Gao, B. Luo, X. Du and M. Guizani, "Security Threats to Hadoop: Data Leakage Attacks and Investigation," *IEEE Network,* vol. 31, pp. 67-71, March-April 2017.

[125] P. Mondal, *Thousands of Unauthenticated Databases Exposed on the Internet,* London, -: RedHunt Labs, 2021.

[126] O. Kolesnikov and H. Parashar, *Detecting Persistent Cloud Infrastructure/Hadoop/YARN Attacks Using Security Analytics: Moanacroner, XBash, and Others,* Jersey City, New Jersey: Securonix Threat Research, 2019.

[127] S. Sinha, S. Gupta and A. Kumar, "Emerging Data Security Solutions in HADOOP based Systems: Vulnerabilities and Their Countermeasures," in *2019 International Conference*

on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 2019.

[128] L. Cheng, Q. Shen and C. Dong, "Invader Job: A Kind of Malicious Failure Job on Hadoop YARN," Kansas City, MO, 2018.

[129] P. Geenens, *Hadoop YARN: An Assessment of the Attack Surface and Its Exploits,* Vols. -, Mahwah, New Jersey: Radware, 2018.

[130] B. Antony, *Secure Communication in Hadoop without Hurting Performance,* Vols. -, San Jose, California: ebay, 2016.

[131] V. R. Bhamidimarri, *Introducing Amazon EMR integration with Apache Ranger,* AWS, 2021.

[132] RedHat, "Red Hat Adds Common Criteria Certification for Red Hat Enterprise Linux 8," RedHat, 2 March 2021. [Online]. Available: https://www.redhat.com/en/about/press-releases/red-hat-adds-common-criteria-certification-red-hat-enterprise-linux-8. [Accessed 27 February 2022].

[133] IEEE Computer Society Center for Secure Design, "Avoiding the Top 10 Software Security Design Flaws," IEEE, Washington, D.C., 2015.

[134] Veracode, "State of Software Security Volume 11," Veracode, Burlington, MA, 2020.

[135] G. Kapil, A. Agrawal, A. Attaallah, A. Algarni, R. Kumar and R. A. Kahn, "Attribute based honey encryption algorithm for securing big data: Hadoop distributed file system perspective," *PeerJ Computer Science,* Vols. -, no. -, 17 February 2020.

[136] Privacera, "Data Governance Immuta vs Apache Ranger: A Failed Benchmark," Privacera, 10 August 2021. [Online]. Available: https://privacera.com/blog/immuta-vs-apache-ranger-a-failed-benchmark/.

[137] TPC, "TPCx-HS Version 2," TPC, 11 April 2022. [Online]. Available: https://www.tpc.org/tpcx-hs/.

[138] Oracle, "Introduction to Oracle Platform Security Services," Oracle, - - -. [Online]. Available: https://docs.oracle.com/cd/E21764_01/core.1111/e10043/underjps.htm#JISEC1827.

[139] Oracle, "The GSS-API: An Overview," Oracle, - - -. [Online]. Available: https://docs.oracle.com/cd/E19683-01/816-1331/overview-6/index.html.