Big Data Working Group

Expanded Top Ten Big Data Security and Privacy Challenges

April 2013

Abstract— Security and privacy issues are magnified by the velocity, volume, and variety of Big Data, such as large-scale cloud infrastructures, diversity of data sources and formats, streaming nature of data acquisition and high volume inter-cloud migration. Therefore, traditional security mechanisms, which are tailored to securing small-scale, static (as opposed to streaming) data, are inadequate. In this paper, we highlight the top ten Big Data security and privacy challenges. Highlighting the challenges will motivate increased focus on fortifying Big Data infrastructures.

Keywords: Big Data; top ten; challenges; security; privacy

Contents

Acknowledgments

CSA Big Data Working Group Co-Chairs

Lead: Sreeranga Rajan, Fujitsu
Co-Chair: Wilco van Ginkel, Verizon
Co-Chair: Neel Sundaresan, eBay

Contributors

Anant Bardhan, CTS
Yu Chen, SUNY Binghamton
Adam Fuchs, Sqrrl
Aditya Kapre
Adrian Lane, Securosis
Rongxing Lu, University of Waterloo
Pratyusa Manadhata, HP Labs
Jesus Molina, Fujitsu
Alvaro Cardenas Mora, University of Texas Dallas
Praveen Murthy, Fujitsu
Arnab Roy, Fujitsu
Shiju Sathyadevan, Amrita University
Nrupak Shah, Dimension Data

CSA Global Staff

Alex Ginsburg, Copyeditor
Luciano JR Santos, Global Research Director
Evan Scoboria, Webmaster
Kendall Scoboria, Graphic Designer
John Yeoh, Research Analyst

Introduction

The term "Big Data" refers to the massive amounts of digital information companies and governments collect about human beings and our environment. The amount of data generated is expected to double every two years, from 2500 exabytes in 2012 to 40,000 exabytes in 2020 [56]. Security and privacy issues are magnified by the volume, variety, and velocity of Big Data. Large-scale cloud infrastructures, diversity of data sources and formats, the streaming nature of data acquisition and high volume inter-cloud migration all create unique security vulnerabilities.

It is not merely the existence of large amounts of data that is creating new security challenges. Big Data has been collected and utilized by many organizations for several decades. The current use of Big Data is novel because organizations of all sizes now have access to Big Data and the means to employ it. In the past, Big Data was limited to very large organizations such as governments and large enterprises that could afford to create and own the infrastructure necessary for hosting and mining large amounts of data. These infrastructures were typically proprietary and were isolated from general networks. Today, Big Data is cheaply and easily accessible to organizations large and small through public cloud infrastructure. Software infrastructures such as Hadoop enable developers to easily leverage thousands of computing nodes to perform data-parallel computing. Combined with the ability to buy computing power on-demand from public cloud providers, such developments greatly accelerate the adoption of Big Data mining methodologies. As a result, new security challenges have arisen from the coupling of Big Data with public cloud environments characterized by heterogeneous compositions of commodity hardware with commodity operating systems, and commodity software infrastructures for storing and computing on data.

As Big Data expands through streaming cloud technology, traditional security mechanisms tailored to securing small-scale, static data on firewalled and semi-isolated networks are inadequate. For example, analytics for anomaly detection would generate too many outliers. Similarly, it is unclear how to retrofit provenance in existing cloud infrastructures. Streaming data demands ultra-fast response times from security and privacy solutions.

The purpose of this paper is to highlight the top ten Big Data security and privacy challenges according to practitioners. To do so, the working group utilized a three-step process to arrive at the top challenges in Big Data:

1. The working group interviewed Cloud Security Alliance (CSA) members and surveyed security-practitioner oriented trade journals to draft an initial list of high priority security and privacy problems.
2. The working group studied published solutions.
3. The working group characterized a problem as a challenge if the proposed solution did not cover the problem scenarios.

Based on this three-step process, the working group compiled the top ten challenges to Big Data security and privacy:

1. Secure computations in distributed programming frameworks
2. Security best practices for non-relational data stores
3. Secure data storage and transactions logs

4. End-point input validation/filtering
5. Real-time security monitoring
6. Scalable and composable privacy-preserving data mining and analytics
7. Cryptographically enforced data centric security
8. Granular access control
9. Granular audits
10. Data provenance

Figure 1 depicts the top ten challenges in the Big Data ecosystem.

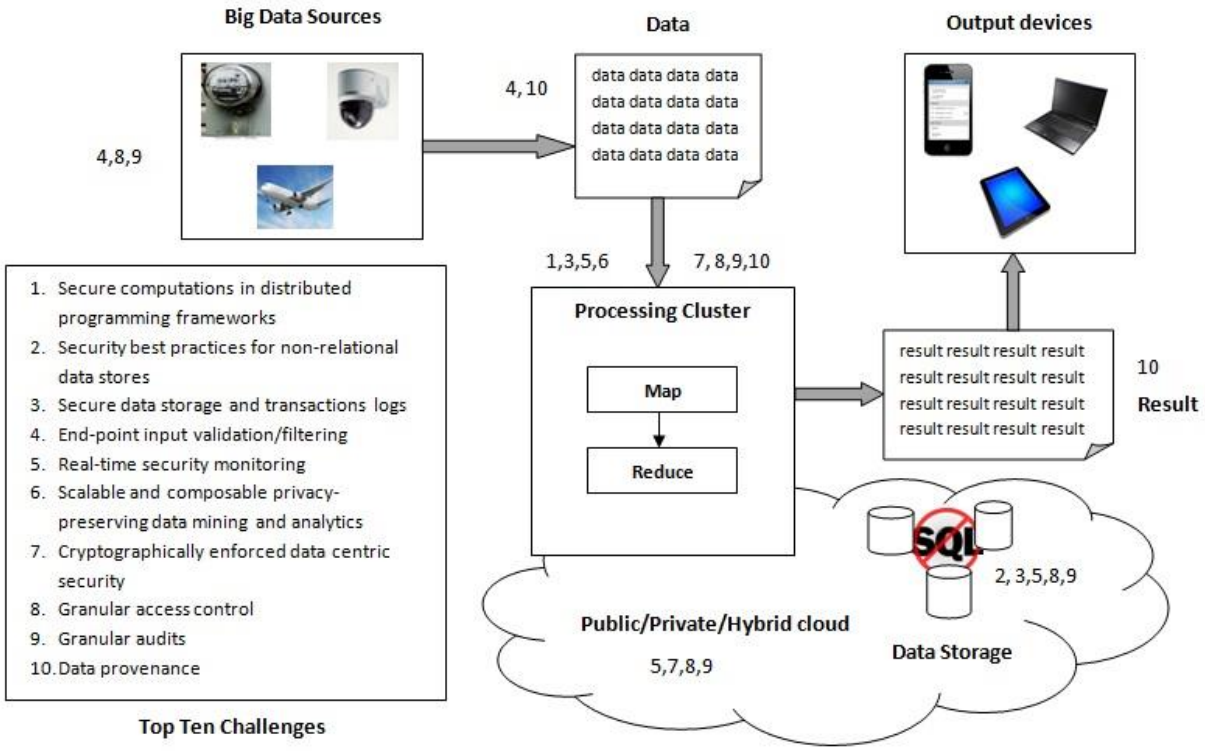

Figure 1: Top Ten Security and Privacy Challenges in the Big Data Ecosystem

The challenges may be organized into four aspects of the Big Data ecosystem, as depicted in Figure 2:

1. Infrastructure Security
2. Data Privacy
3. Data Management
4. Integrity and Reactive Security

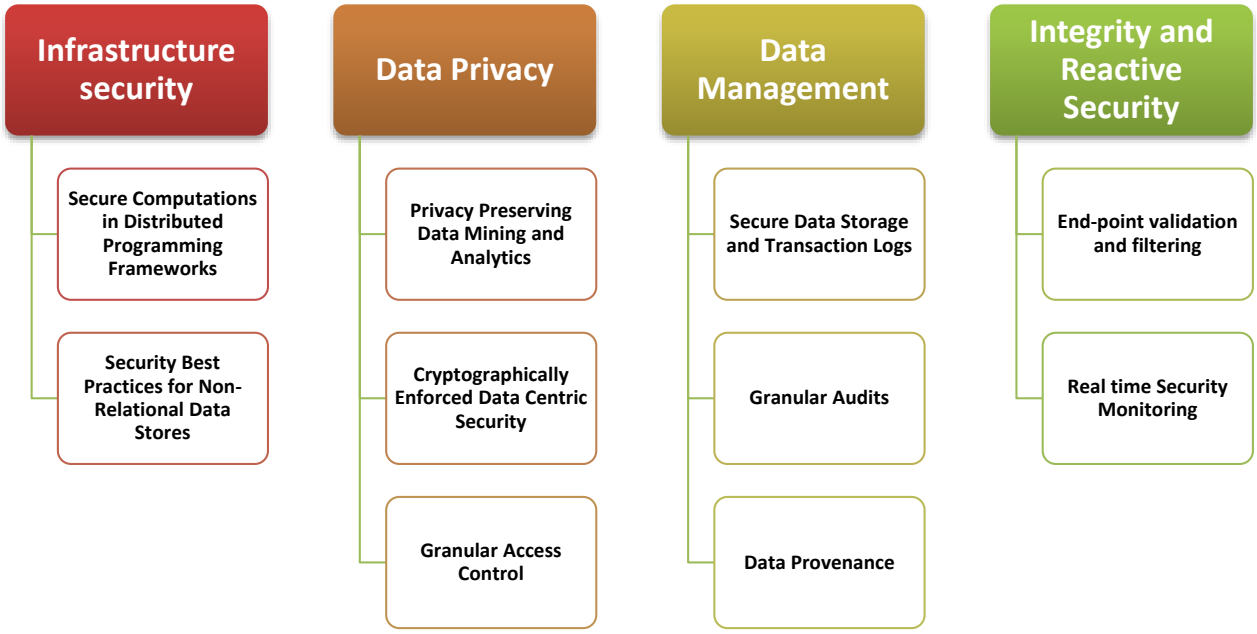

Figure 2: Classification of the Top 10 Challenges

In order to secure the infrastructure of Big Data systems, the distributed computations and data stores must be secured. To secure the data itself, information dissemination must be privacy-preserving, and sensitive data must be protected through the use of cryptography and granular access control. Managing the enormous volume of data necessitates scalable and distributed solutions for both securing data stores and enabling efficient audits and data provenance. Finally, the streaming data emerging from diverse end-points must be checked for integrity and can be used to perform real time analytics for security incidents to ensure the health of the infrastructure.

Solving security and privacy challenges typically requires addressing three distinct issues:

1. Modeling: formalizing a threat model that covers most of the cyber-attack or data-leakage scenarios
2. Analysis: finding tractable solutions based on the threat model
3. Implementation: implementing the solution in existing infrastructures

In this paper, we provide a brief description of each challenge, review usage of Big Data that may be vulnerable, and summarize existing knowledge according to the modeling, analysis, and implementation for each challenge.

1.0 Secure Computations in Distributed Programming Frameworks

Distributed programming frameworks utilize parallel computation and storage to process massive amounts of data. For example, the MapReduce framework splits an input file into multiple chunks. In the first phase of MapReduce, a Mapper for each chunk reads the data, performs some computation, and outputs a list of key/value pairs. In the next phase, a Reducer combines the values belonging to each distinct key and outputs the result. There are two major attack prevention measures: securing the mappers and securing the data in the presence of an untrusted mapper.

1.1 Use Case

Untrusted mappers can be altered to snoop on requests, alter MapReduce scripts, or alter results. The most difficult problem is to detect mappers returning incorrect results, which will, in turn, generate incorrect aggregate outputs. With large data sets, it is nearly impossible to identify malicious mappers that may create significant damage, especially for scientific and financial computations.

Retailer consumer data is often analyzed by marketing agencies for targeted advertising or customer-segmenting. These tasks involve highly parallel computations over large data sets and are particularly suited for MapReduce frameworks such as Hadoop. However, the data mappers may contain intentional or unintentional leakages. For example, a mapper may emit a unique value by analyzing a private record, undermining users' privacy.

1.2 Modeling

The threat model for mappers has three major scenarios:

1. Malfunctioning Compute Worker Nodes – Workers assigned to mappers in a distributed computation could malfunction due to incorrect configuration or a faulty node. A malfunctioning Worker could return incorrect output from the mapper, which may compromise the integrity of the aggregate result. Such a Worker may also be modified to leak users' confidential data or profile users' behaviors or preferences for privacy mining.
2. Infrastructure Attacks – Compromised Worker nodes may tap the communication among other Workers and the Master with the objective of replay, Man-In-the-Middle, and DoS attacks to the MapReduce computations.
3. Rogue Data Nodes – Rogue data nodes can be added to a cluster, and subsequently receive replicated data or deliver altered MapReduce code. The ability to create snapshots of legitimate nodes and re-introduce altered copies is a straightforward attack in cloud and virtual environments and is difficult to detect.

1.3 Analysis

Based on the threat model outlined above, there are two dimensions of analysis: ensuring trustworthiness of mappers and securing the data despite untrusted mappers.

For ensuring the trustworthiness of mappers, there are two techniques: trust establishment and Mandatory Access Control (MAC) [1].

1. Trust establishment has two steps: initial trust establishment followed by periodic trust update. When a Worker sends a connection request to the Master, the Master authenticates the Worker. Only authenticated Workers with expected properties will be assigned a mapper task. Following the initial authentication, the security properties of each Worker are checked periodically for conformance with predefined security policies.
2. MACensures access to the files authorized by a predefined security policy. MAC ensures integrity of inputs to the mappers, but does not prevent data leakage from the mapper outputs.

In order to prevent information leakage from mapper outputs, data de-identification techniques are required to prevent violation of privacy through the output of aggregate computations. A mathematically rigorous definition for data de-identification is the notion of differential privacy, which is achieved by adding random noise to the output of a computation. However, it is difficult to prove that a particular technique is privacy-preserving.

1.4 Implementation

MAC is implemented in Airavat [1] by modifying the MapReduce framework, the distributed file system, and the Java virtual machine with SELinux as the underlying operating system. MAC in SELinux ensures that untrusted code does not leak information via system resources. However it cannot guarantee privacy for computations based on output keys produced by untrusted mappers. To prevent information leakage through the outputs, it relies on a recently developed de-identification framework of differential privacy based on function sensitivity. In the context of mappers, function sensitivity is the degree of influence that an input can have the mapper output. Estimating the sensitivity of arbitrary untrusted code is difficult.

There are two problems to be tackled with the solutions outlined above for widespread practical adoption:

1. Performance penalties due to imposing MAC
2. Limitations of differential privacy in providing guarantees

2.0 Security Best Practices for Non-Relational Data Stores

The security infrastructures of non-relational data stores popularized by NoSQL databases are still evolving [2]. For instance, robust solutions to NoSQL injection are still not mature. Each NoSQL database was built to tackle different challenges posed by the analytics world, and security was never addressed during the design stage. Developers using NoSQL databases usually embed security in the middleware. NoSQL databases do not provide any support for explicitly enforcing security in the database. However, clustering aspects of NoSQL databases pose additional challenges to the robustness of such security practices.

2.1 Use Case

Companies dealing with large unstructured data sets may benefit by migrating from a traditional relational database (RDB) to a NoSQL database. NoSQL databases accommodate and process huge volumes of static and streaming data for predictive analytics or historical analysis. Threat trees derived from detailed threat analysis using threat-modeling techniques on widely used NoSQL databases demonstrate that NoSQL databases only have a very thin security layer, compared to traditional RDBs. In general, the security philosophy of NoSQL databases relies on external enforcement mechanisms. To reduce security incidents, the company must review security policies for the middleware and, at the same time, toughen the NoSQL database itself to match the security RDBs without compromising on its operational features. The capability of NoSQL databases to perform analytics over unstructured and structured data at ease is in no way comparable to an RDB's ability to handle OLTP and OLAP (to a large extent with the latest RDBMS versions). However, it is important that security loopholes within NoSQL databases are plugged without compromising on its outstanding analytical capabilities.

Cloud-based solutions, in which traditional service providers offer Analytics as a Service (AaaS), are based on analytics frameworks built using a combination of tools capable of handling both streaming and static data, with NoSQL databases used for intermediate data handling. In such scenarios, several users share the framework, feeding both streaming and static data with appropriate connectors through the framework for analytics. These data sets need to be held in a NoSQL database for intermediate processing before the results are pushed to the respective users. With current NoSQL security mechanisms, it is virtually impossible to segregate sensitive data pertaining to different cloud users sharing the framework's internal NoSQL database.

2.2 Modeling

The same architectural flexibility that allows the two notable features of NoSQL, performance and scalability, poses the greatest security risk [3]. NoSQL was designed with the vision of tackling large data sets, with limited emphasis on security [4]. This has caused many critical security flaws in NoSQL, only a few of which are addressed in this paper. Lack of security standards has caused vendors to develop bottom-up NoSQL solutions and address security issues on an ad-hoc basis. The threat model of NoSQL databases has six major scenarios:

1. Transactional Integrity – One of the most visible drawbacks of NoSQL is its soft approach towards ensuring transactional integrity. Introducing complex integrity constraints into its architecture will fail NoSQL's primary objective of attaining better performance and scalability. Techniques like Architectural Trade-off Analysis Method (ATAM) specifically deal with the trade-offs in quality requirements in architectural decision (for example, performance vs. security). This analytical method can be utilized to evaluate the level of integrity constraints that may be infused into a core architectural kernel without significantly affecting performance.
2. Lax Authentication Mechanisms – Across the board, NoSQL uses weak authentication techniques and weak password storage mechanisms. This exposes NoSQL to replay attacks and password brute force attacks, resulting in information leakage. NoSQL uses HTTP Basic- or Digest-based authentication, which are prone to replay or man-in-the-middle attack. REST, which is another preferred communication protocol, is also based on HTTP and is prone to cross-site scripting, cross-site request forgery, injection attacks, etc. Above all, NoSQL does not support integrating third-party pluggable modules to enforce authentication. By manipulating the RESTful connection definition, it is possible to get access to the handles and configuration parameters of the underlying database, thereby gaining access to the file system. Although some of the existing NoSQL databases offer authentication at the local node level, they fail to enforce authentication across all the cluster nodes.
3. Inefficient Authorization Mechanisms – Authorization techniques differ from one NoSQL solution to another. Most of the popular solutions apply authorization at higher layers rather than enforcing authorization at lower layers. More specifically, authorization is enforced on a per-database level rather than at the collection level. There is no role-based access control (RBAC) mechanism built into the architecture because defining user roles and security groups with an RBAC mechanism is impossible.
4. Susceptibility to Injection Attacks – Easy to employ injection techniques allow backdoor access to the file system for malicious activities. Since NoSQL architecture employs lightweight protocols and mechanisms that are loosely coupled, it is susceptible to various injection attacks like JSON injection, array injection, view injection, REST injection, GQL injection, schema injection, etc. For example, an attacker can utilize schema injection to inject thousands of columns onto the database with data of the attacker's choice. The impact of such an attack can range from a database with corrupted data to a DoS attack resulting in total unavailability of the database.
5. Lack of Consistency – The inability to simultaneously enforce all three elements of the CAP theorem (consistency, availability, and partition tolerance) while in distributed mode undermines the trustworthiness of the churned results. As a result, users are not guaranteed consistent results at any given time, as each participating node may not be entirely synchronized with the node holding the latest image. Current hashing algorithms entrusted to replicate data across the cluster nodes crumple in the event of a single node failure, resulting in load imbalance among the cluster nodes.
6. Insider Attacks - Lenient security mechanisms can be leveraged to achieve insider attacks. These attacks could remain unnoticed because of poor logging and log analysis methods, along with other rudimentary security mechanisms. As critical data is stowed away under a thin security layer, it is difficult to ensure that the data owners maintain control.

2.3 Analysis

Data integrity needs to be enforced through an application or middleware layer. Passwords should never be left in the clear while at rest and in transit, but instead should be encrypted or hashed using secure hashing algorithms. Similarly, data stored in the database should never be left in the clear. Considering the already weak authentication and authorization techniques employed, it is vital to keep the data encrypted while at rest despite the associated performance impacts. Hardware appliance-based encryption/decryption and bulk file-based encryption are faster and would alleviate some concern about the performance impacts of encryption. Of course, hardware-based encryption is not without its own criticism as it often leads to a vendor lock-in, low-strength key used in encryption/decryption that can be exploited by attackers. As a result, malicious users who gain access to the file system could directly extract sensitive data from the file system. In order to maintain confidentiality while in transit, it is good practice to use SSL/TLS to establish connections between the client and the server and also for communication across participating cluster nodes. Adopting such mechanisms to exchange mutually verifiable keys and establish trust would ensure data confidentiality while data is in transit. The NoSQL architecture should support pluggable authentication modules with the capacity to enforce security at all levels as the situation demands.

Communication across clusters should also be better controlled so that each node can validate the trust level of other participating nodes before establishing a trusted communication channel. The utilization of intelligent hashing algorithms can ensure that the data is replicated consistently across nodes, even during node failures. All NoSQL products/solutions recommend that they be run on a trusted environment, which ensures that only trusted machines can access the database ports.

Appropriate logging mechanisms, on-the-fly log analysis, and aggregation and application of correlation to log analysis could expose possible attacks. Applying fuzzing methods (providing invalid, unexpected or random inputs) can be an ideal method to expose possible vulnerabilities in NoSQL that engages HTTP to establish communication with the clients.

Appropriate data tagging techniques, with time stamps enforced through intelligent algorithms while piping data from its source, will defend against unauthorized modification of data. These techniques will also preserve the authenticity of the data stored.

2.4 Implementation

It is evident from the threat models and analysis that plugging NoSQL holes by wrapping sensitive data within the thin, easily penetrable security layer of web or similar interfaces is insufficient to guarantee the security of the underlying data [5]. Hiding NoSQL under the secure wrapper of middleware or accessing NoSQL using a framework like Hadoop can create a virtual secure layer around the NoSQL perimeter. Object-level security at the collection- or column-level can be induced through the middleware, retaining its thin database layer. Such a methodology will ensure that there is no direct access to the data and that the data is only exposed based on the controls configured within the middleware or framework layer. Cloud-era distributed Hadoop supports strong Kerberos authentication. Such mechanisms could:

1. Prevent malicious user impersonation
2. Enforce user authentication on all remote procedure calls (RPCs)
3. Prevent user manipulation of group membership by performing group resolution on the Hadoopmaster nodes, cluster nodes and job tracker
4. Ensure proper isolation as map tasks are run under the account of the user who submitted the job.

Deploying the middleware layer to encapsulate the underlying NoSQL stratum can be another option to implement security. Most of the middleware software comes with ready-made support for authentication, authorization and access control. In the case of Java, Java Authentication and Authorization Services (JAAS) and SpringSource, Spring Security frameworks are deployed for authentication, authorization and access control. Such an architecture will ensure that any changes to schema, objects and/or data are validated, thus gaining better control while preserving the capabilities of NoSQL [6].

In order to maintain performance, the ability to scale to demand, and the security of the overall system, it is necessary to integrate NoSQL into a framework, thereby offloading the security components onto the framework. Such a framework should be tightly coupled with the underlying operating system so that policy-based security layers can be burned onto the lower substratum (kernel layer). This will ensure that the missing RBAC can be enforced in order to limit access to the underlying data, preserve the thinness of the database layer, and maintain the analytical capability of NoSQL. Above all, such a mechanism will ensure that the data owners have better control over their data, thus preventing or exposing insider attacks. This will tighten the security of data intermediately held for processing in a NoSQL database within a shared framework architecture in which analytics is offered as a service through one or more cloud service models.

As an alternative to vulnerable NoSQL data, encryption provides better protection. Hadoop employs file-layer encryption to provide unswerving protection, irrespective of the operating system, platform or storage type. With the availability of products capable of offering encryption, demand for handling streaming data and processing them in-memory is on the rise. Encryption solutions seem to be a cost effective way to address several of the known data-security issues.

3.0 Secure Data Storage and Transactions Logs

Data and transaction logs are stored in multi-tiered storage media. Manually moving data between tiers gives the IT manager direct control over exactly what data is moved and when. However, as the size of data set continues to grow exponentially, scalability and availability have necessitated auto-tiering for Big Data storage management. Auto-tiering solutions do not keep track of where the data is stored, which poses new challenges to secure data storage. New mechanisms are imperative to thwart unauthorized access and maintain constant availability.

3.1 Use Case

A manufacturer wants to integrate data from different divisions. Some of this data is rarely retrieved, while some divisions constantly utilize the same data pools. An auto-tier storage system will save the manufacturer money by pulling the rarely utilized data to a lower (and cheaper) tier. However, this data may consist of critical information that is not regularly accessed (such as R&D results). Since the lower-tier often provides reduced security, the company should carefully study tiering strategies. Additionally, the meta-data, i.e. the text log, introduces another dimension that needs to be protected. The log poisoning attacks will potentially lead to data inconsistency and disputes among users.

3.2 Modeling

Network-based, distributed, auto-tier storage systems are a promising solution that possess advanced features such as transparent service, good scalability and elasticity. However, auto-tier storage systems generate new vulnerabilities due to a lack of physical possession, untrusted storage service, or inconsistent security policies. The threat model for auto-tier storage systems includes seven major scenarios:

1. Confidentiality and Integrity – In addition to those attempting to steal sensitive information or damage user data, storage service providers are also assumed to be untrustworthy third parties. Data transmission among tiers in a storage system provides clues that enable the service provider to correlate user activities and data set. Without being able to break the cipher, certain properties can be revealed.
2. Provenance – Due to the extremely large size, it is infeasible to download the entire data set to verify its availability and integrity. Lightweight schemes are desired to provide verification that is probabilistically accurate and incurs low computing and communication overhead.
3. Availability – Auto-tiering also places challenges on the service providers to guarantee constant availability. Not only does weaker security at lower tiers risk Denial of Service (DoS) attacks, the performance gap between lower tiers and higher tiers also extends the backup windows during periods of fast restore and disaster recovery.
4. Consistency – It is now typical that data flows among tiers and is shared by multiple users. To maintain consistency among multiple duplicates stored at difference locations is non-trivial. Two issues that need to be addressed carefully are write-serializability and multi-writer multi-reader (MWMR) problems.

5. Collusion Attacks – While a data owner stores the cipher text in an auto-tier storage system and distributes the key and permission access to the users, each user is authorized to have access to a certain portion of the data set. Also, the service provider cannot interpret the data without the cipher key materials. However, if the service provider colludes with users by exchanging the key and data, they will obtain a data set that they are not entitled to.
6. Roll-Back Attacks – In a multi-user environment, the service provider can launch roll-back attacks on users. When an updated version of a data set has been uploaded into storage, the service provider can fool the user by delivering the outdated version. Certain evidence is required to help users ensure that data is up-to-date, and the user should have the capability of detecting the inconsistency. This is also referred to as "User's Freshness."
7. Disputes – A lack of recordkeeping will lead to disputes between users and the storage service provider, or among users. When data loss or tampering occurs, transmission logs/records are critical to determining responsibility. For example, a malicious user outsources data to a storage system. Later, the user reports data loss and asks for compensation for his claimed loss. In this case, a well-maintained log can effectively prevent fraud.

3.3 Analysis

In recent decades, the information assurance and cyber infrastructure security fields have developed rapidly. There are now sophisticated techniques to address the security issues listed above. Confidentiality and integrity can be achieved with robust encryption techniques and message-digests. The exchange of signed message-digests can be used to address potential disputes [18]. User freshness and write-serializability can be solved by periodic audit [24] and chain hash [23] or persistent authenticated dictionary (PAD) [11]. Secure untrusted data repository (SUNDR) can be used to detect fork consistency attack and write serializability.

Two "lock-free" protocols, linear and concurrent, have been proposed to address the problem of single-write multi-read (SWMR) [21]. However, SWMR is a case-dependent problem and is beyond the scope of this paper. Broadcast encryption [14] and key rotation [20] can be used to improve scalability. Researchers have proposed technologies to handle the provenance issues [22]. Data availability can be improved through proof of retrievability (POR) or provable data possession (PDP) methods with high probability [10], [19].

Regarding collusion attacks, as long as the users do not exchange their private keys, a policy-based encryption system (PBES) can successfully guarantee a collusion-free environment [13]. If the users are willing to exchange their private keys without exchanging the decrypted content, a mediated decryption system can avoid collusion attacks. If the users are willing to exchange the decrypted contents, digital rights management can prevent collusion attacks. Two non-repudiation protocols have been proposed recently to address disputed issues [16], [17].

While it seems that there are techniques for each individual security problem in large scale auto-tier storage systems, there is no systematic approach to integrate them into a seamless, holistic solution. The non-uniform security policies among different tiers pose an additional challenge to securing inter-tier data transmission. More considerations are necessary to balance tradeoffs among security, usability, complexity, and cost.

3.4 Implementation

The heterogeneity of technologies, varied security policies, and cost constraints lead to diverse security strategies in multitier storage systems. While many structures can be adopted to satisfy general security requirements such as data confidentiality, integrity, and availability, three special issues need more attention:

1. Dynamic Data Operations – Data set in the auto-tier storage system is dynamic because operations such as modification, duplication, deletion and insertion will happen more frequently. An extended dynamic version of a PDP scheme [12] achieves higher efficiency because it only relies on symmetry-key cryptography. However, since the number of queries is limited, the scheme cannot support fully dynamic data operations. A formal framework for dynamic provable data possession (DPDP) [15] can be considered to improve detection probability at the cost of increased server computation burden. An extended version of the compact POR [26] supports both public verifiability and data dynamics in cloud storage, which can be thought of as a special case of a network-based auto-tier storage system.
2. Privacy Preservation – It is a trend to outsource verification procedures to a third-party auditor (TPA), and the verification protocol is expected to be publicly verifiable. At first glance, privacy preservation seems to be in conflict with the public verifiability requirement. A privacy-preserving public auditing scheme was proposed for cloud storage in Wang, et al. [25]. Based on a homomorphic linear authenticator integrated with random masking, the proposed scheme is able to preserve data privacy when a TPA audits the data set stored in the servers at different tiers.
3. Secure Manipulations on Encrypted Data – Beyond privacy-preserving auditing, today's computing task outsourcing requires the capacity to conduct operations on the cipher-text without decryption. The fully homomorphic encryption scheme makes these operations possible because more complex functions are supported [54]. Recent achievements in "Cryptographic Cloud Storage" [55] provide an alternative solution for cloud platforms by allowing the construction of a secure IaaS storage on top of an untrusted infrastructure.

4.0 End-Point Input Validation/Filtering

Many Big Data uses in enterprise settings require data collection from a variety of sources, including end-point devices. For example, a security information and event management system (SIEM) may collect event logs from millions of hardware devices and software applications in an enterprise network. A key challenge in the data collection process is input validation: how can we trust the data? How can we validate that a source of input data is not malicious? And how can we filter malicious input from our collection? Input validation and filtering is a daunting challenge posed by untrusted input sources, especially with the bring-your-own-device (BYOD) model.

4.1 Use Case

Data retrieved from weather sensors and feedback votes sent by an iPhone application share a similar validation problem. A motivated adversary may be able to create "rogue" virtual sensors or spoof iPhone IDs to rig the results. This is further complicated by the amount of data collected, which may exceed millions of readings or votes. To perform these tasks effectively, algorithms need to be created to validate the input for large data sets.

4.2 Modeling

A threat model for input validation has four major scenarios:

1. An adversary may tamper with a device from which data is collected, or may tamper with the data collection application running on the device to provide malicious input to a central data collection system. For example, in the case of iPhone feedback voting, an adversary may compromise an iPhone (i.e., the iPhone's software platform) or may compromise the iPhone app that collects user feedback.
2. An adversary may perform ID cloning attacks (e.g., Sybil attacks) on a data collection system by creating multiple fake identities (e.g., spoofed iPhone IDs) and by then providing malicious input from the faked identities. The challenges of Sybil attacks become more acute in a BYOD scenario. Since an enterprise's users are allowed to bring their own devices and use them inside the enterprise network, an adversary may use her device to fake the identity of a trusted device and then provide malicious input to the central data collection system.
3. A more complicated scenario involves an adversary that can manipulate the input sources of sensed data. For example, instead of compromising a temperature sensor, an adversary may be able to artificially change the temperature in a sensed location and introduce malicious input to the temperature collection process. Similarly, instead of compromising an iPhone or a GPS-based location sensing app running on the iPhone, an adversary may compromise the GPS signal itself by using GPS satellite simulators [7].
4. An adversary may compromise data in transmission from a benign source to the central collection system (e.g., by performing a man-in-the-middle attack or a replay attack). This issue will be discussed in detail in Section 7.

4.3 Analysis

Given the above threat model, solutions to the input validation problem fall into two categories: (a) solutions that prevent an adversary from generating and sending malicious input to the central collection system, and (b) solutions that detect and filter malicious input at the central system if an adversary successfully inputs malicious data.

Preventing an adversary from sending malicious input requires tamper-proof software and defenses against Sybil attacks. Research on the design and implementation of tamper-proof secure software has a very long history in both academia and industry. Many design and development tools, techniques, and best practices have been developed to identify and remove vulnerabilities from software. However, developing complex software devoid of vulnerabilities is nearly impossible. Moreover, though security of PC-based software platforms and applications has been widely studied, mobile device and application security remains an active area of research. As a result, we assume that a determined adversary will be able to compromise mobile devices and the applications running on them. Gilbert, et al. recently proposed to use Trusted Platform Modules (TPMs) to guarantee the integrity of raw sensor data, as well as data derived from raw data [8]. TPMs, however, are not universally found in mobile devices. Moreover, even in the presence of TPMs, an adversary can manipulate the sensor inputs (e.g., GPS signals).

Defense schemes against ID cloning attacks and Sybil attacks have been proposed in diverse areas such as peer-to-peer systems, recommender systems, vehicular networks, and wireless sensor networks [9]. Many of these schemes propose to use trusted certificates and trusted devices to prevent Sybil attacks. However, managing certificates in a large enterprise setting with millions of entities is challenging. Many other schemes have proposed variations of the resource testing idea, such as determining if multiple fake identities possess fewer resources than expected from independent genuine identities. Resource testing provides minimal defense against Sybil attacks by discouraging Sybil attacks instead of preventing them.

The strengths of Big Data can be utilized to detect and filter malicious inputs at the central collection system. Since a real-world data collection system collects large amounts of data from millions of sources, malicious input from an adversary may appear as outliers. Therefore, existing statistical similarity detection techniques and outlier detection techniques may be employed to detect and filter out malicious input.

4.4 Implementation

There is no foolproof approach for input validation and filtering. As a result, we recommend a hybrid approach to be implemented in practice. First, Big Data collection system designers should take utmost care to develop secure data collection platforms and applications. They should especially consider the BYOD scenario in which their application would run on untrusted devices. Second, designers should identify plausible Sybil attacks and ID spoofing attacks on their system and then identify cost-effective ways to mitigate the attacks. Third, designers should acknowledge that a determined adversary will be able to send malicious input to their central collection system. In response, designers should develop algorithms to detect and filter out malicious input from an adversary.

5.0 Real-Time Security Monitoring

Big Data and security do not only intersect at the protection of Big Data infrastructures, but also at the leveraging of Big Data analytics to help improve the security of other systems.

One of the most challenging Big Data analytics problems is real-time security monitoring, which consists of two main angles: (a) monitoring the Big Data infrastructure itself and (b) using the same infrastructure for data analytics. An example of (a) is the monitoring of the performance and health of all the nodes that make up the Big Data infrastructure. An example of (b) would be a health care provider using monitoring tools to look for fraudulent claims or a cloud provider using similar Big Data tools to get better real-time alert and compliance monitoring. These improvements could provide a reduction in the number of false positives and/or an increase in the quality of the true positives. In this paper, we focus on both angles.

Real-time security monitoring is a challenge because of the number of alerts generated by security devices. These alerts (correlated or not) lead to a massive number of false positives, which are often ignored due to limited human capacity for analysis. This problem might even increase with Big Data, given the volume and velocity of data streams. However, Big Data technologies may provide an opportunity to rapidly process and analyze different types of data. These technologies can be used to provide, for instance, real-time anomaly detection based on scalable security analytics.

5.1 Use Case

Most industries and government agencies will benefit from real-time security analytics, although specific usage may vary. Common uses include utilizing the technology to answer questions such as, “Who is accessing which data from which resource at what time,” “Are we under attack,” or “Is there a breach of compliance standard C because of action A?” These analytic areas are not new, but improved data collection and analytics allows for faster and better decisions (i.e., less false positives). In addition to these improved analytic areas, new uses can be defined, or we can redefine existing uses vis-à-vis Big Data.

For example, the health industry benefits greatly from Big Data technologies, potentially saving billions to the taxpayer by becoming more accurate with the payment of claims and reducing the fraud related to claims. At the same time, stored records are extremely sensitive and must comply with patient privacy regulations. As a result, healthcare data must be carefully protected. Real-time detection of the anomalous retrieval of personal information, intentional or unintentional, allows the healthcare provider to quickly repair damage and prevent further misuse.

5.2 Modeling

Security monitoring requires that the Big Data infrastructure, or platform, is inherently secure. Threats to a Big Data infrastructure include rogue admin access to applications or nodes, (web) application threats, and eavesdropping on the line. Such an infrastructure is mostly an ecosystem of different components, where (a) the

security of each component and (b) the security integration of these components must be considered. For example, if we run a Hadoop cluster in a public cloud, one has to consider:

1. The security of the public cloud, which itself is an ecosystem of components consisting of computing, storage and network components.
2. The security of the Hadoop cluster, the security of the nodes, the interconnection of the nodes, and the security of the data stored on a node.
3. The security of the monitoring application itself, including applicable correlation rules, which should follow secure coding principles and best practices.
4. The security of the input sources (e.g., devices, sensors) that the data comes from.

The other major threat model revolves around adversaries that will try to evade the Big Data analytics tools used to identify them. Attackers can create evasion attacks [51] in an effort to prevent being detected, and also launch data poisoning attacks [52] to reduce the trustworthiness of the data sets used to train Big Data analytics algorithms.

Apart from these security threats, other barriers become important, such as legal regulations. Depending on where the monitored data exists, certain privacy laws may apply. This may create hurdles because some data might not be available for security monitoring or may only be available in a certain format (e.g., anonymized).

5.3 Analysis

Big Data analytics can be used to monitor anomalous connections to the cluster and mine logging events to identify suspicious activities.

People implementing mining and analytics algorithms should keep in mind the adversarial problem of the statistics they are computing in order to mitigate potential evasion or poisoning attacks. In addition, there is still much debate around the legal and ethical aspects of using Big Data in analytics. Monitoring Big Data is a combination of different factors (e.g., technical, legal, and ethical) that must be taken into account. The next section will focus on the privacy-preserving mechanisms we can incorporate into Big Data analytics systems to improve data management.

5.4 Implementation

The implementation of the security best practices depends on the situation at hand. At the moment of writing, there are no built-in security monitoring and analysis tools in Hadoop. However, monitoring and analysis tools are being developed and announced by different Hadoop providers and vendors. An alternative solution is the implementation of front-end systems to monitor Hadoop requests (e.g., Database Activity Monitoring proxy of firewall). The application security depends on the application itself and whether security controls have been built-in (i.e., adhere to OWASP guidelines). With respect to real-time monitoring, solutions and frameworks for real-time monitoring (like NIST's Security Content Automation Protocol (SCAP)) are slowly entering the Big Data arena. One of the criticisms of these tools in Hadoop is that they are only batch-oriented, which is useful for historical or

trend analysis, but not for real-time monitoring. Examples of attempts to overcome this hurdle include Storm (storm-project.net) and Apache Kafka. Other real-time streaming applications, which are built upon Hadoop, are now entering the market.

6.0 Scalable and Composable Privacy-Preserving Data Mining and Analytics

As described by Boyd and Crawford [27], Big Data can potentially enable invasions of privacy, invasive marketing, decreased civil liberties, and increased state and corporate control.

A recent analysis of how companies are leveraging data analytics for marketing purposes included an example of how a retailer was able to identify a teen's pregnancy before her father learned of it [28]. Similarly, anonymizing data for analytics is not enough to maintain user privacy. For example, AOL released anonymized search logs for academic purposes, but users were easily identified by their searches [29]. Netflix faced a similar problem when anonymized users in their data set were identified by correlating Netflix movie scores with IMDB scores.

Therefore, it is important to establish guidelines and recommendations for preventing inadvertent privacy disclosures.

6.1 Use Case

User data collected by large organizations is constantly accessed by inside analysts as well as outside contractors and business partners. A malicious insider or untrusted partner can abuse these data sets and extract private information from customers.

Similarly, intelligence agencies require the collection of vast amounts of data. The data sources are numerous and may include chat-rooms, personal blogs and network routers. However, most data is innocent in nature and does not to be retained, thereby preserving anonymity. Robust and scalable privacy-preserving mining algorithms increase the chances of collecting relevant information to increase everyone's safety.

6.2 Modeling

Multiple threat models can compromise user privacy in Big Data stores. A malicious attacker can compromise the data store by exploiting vulnerabilities in the company hosting the data (for example, the Lulzsec hacking group obtained data from multiple sites, including HBGary Federal). A threat model for user privacy shows three major scenarios:

1. An insider in the company hosting the Big Data store can abuse her level of access and violate privacy policies. An example of this scenario is the case of a Google employee who stalked teenagers by monitoring their Google chat communications [30].
2. If the party owning the data outsources data analytics, an untrusted partner might be able to abuse their access to the data to infer private information from users. This case can apply to the usage of Big Data in the cloud, as the cloud infrastructure (where data is stored and processed) is not usually controlled by the owners of the data.

3. Sharing data for research is another important use. However, as we pointed out in the introduction to this section, ensuring that the data released is fully anonymous is challenging because of re-identification. EPIC's definition of re-identification is the process by which anonymized personal data is matched with its true owner. Several examples of re-identification can be seen in EPIC's website [31].

6.3 Analysis

In order to protect the user privacy, best practices in the prevention and detection of abuse by continuous monitoring must be implemented. Privacy-preserving analytics is an open area of research that can help minimize the success of malicious actors from the data set. However, there are few practical solutions at the moment.

Differential privacy is a good first step toward privacy preservation. Differential privacy defines a formal model of privacy that can be implemented and proven secure at the cost of adding computational overhead and noisy results to data analytics results. Perhaps the current definition of differential privacy is too conservative, and a new more practical definition might address some of costs associated with the implementation of this principle.

Another potential solution to outsourced computational resources is universal homomorphic encryption, which promises to provide data analytics while the outsourced data remains encrypted. This technology is currently in its infancy and is not practical for current deployments, but it is a promising field for long-term research.

Privacy must be preserved under composition. In other words, leakage of private information is controlled even if multiple databases are linked. Linking anonymized data stores is challenging because we need to maintain consistency between anonymized data.

6.4 Implementation

Some basic technical implementation principles to prevent attacks from malicious outsiders include the encryption of data at rest, access control, and authorization mechanisms. It is also important to keep software infrastructure patched with up-to-date security solutions in order to minimize exploitable vulnerabilities.

To minimize the potential abuse of insiders, Big Data operators need to design their system following the separation of duty principle, which would force malicious insiders to collude. In addition, a clear policy for logging access to the data set can help with forensics and act as a deterrent, informing potential malicious insiders that their activities can be traced.

Data sharing in the context of privacy protection is currently an open area of research. A best-practice recommendation for this scenario is to be aware of re-identification techniques and to know that anonymization is not enough to guarantee privacy.

While this paper is focused on technical analysis, implementations must also follow user privacy regulations. For example, European Union countries are regulated by Directives 95/46/EC on the protection of personal data, 2002/58/EC on Privacy and Electronic Communications, and 2006/24/EC on the retention of data generated or processed in electronic communication. Similarly in the United States, access to stored data is regulated by the

Electronic Communications and Privacy Act, and streaming analytics is regulated by the U.S.A. PATRIOT Act and the Wiretap Act. More information on policy, law enforcement and their relation to privacy can be found in Diffie and Landau's publication [53].

7.0 Cryptographically Enforced Data-Centric Security

There are two fundamentally different approaches to controlling the visibility of data to different entities, such as individuals, organizations and systems. The first approach controls the visibility of data by limiting access to the underlying system, such as the operating system or the hypervisor. The second approach encapsulates the data itself in a protective shell using cryptography. Both approaches have their benefits and detriments. Historically, the first approach has been simpler to implement and, when combined with cryptographically-protected communication, is the standard for the majority of computing and communication infrastructure.

However, the system-based approach arguably exposes a much larger attack surface. The literature on system security is replete with attacks on the underlying systems to circumvent access control implementations (such as buffer overflow and privilege escalation) and access the data directly. On the other hand, protecting data end-to-end through encryption exposes a smaller, more well-defined attack surface. Although covert side-channel attacks [36], [37] are possible to extract secret keys, these attacks are far more difficult to mount and require sanitized environments.

7.1 Use Case

Since Big Data comes from diverse end-points and contains more personal data, it is becoming increasingly essential to tether the visibility of the data at the source. Legal frameworks like the Health Insurance Portability and Accountability Act (HIPAA) can only help implicate responsible parties after sensitive data is compromised. At that point, the damage is already done.

This raises an immediate problem of indexing, analyzing and meaningfully processing the encrypted data. Complementary to the issues of data confidentiality, the integrity of data also has to be ensured to prevent data poisoning, especially for data sets with diverse sources.

7.2 Modeling

Threat models in cryptography are mathematically defined through an interaction between systems implementing the protocol and an adversary, which is able to access the externally visible communications and to compute any probabilistic polynomial time function of the input parameters. The system is deemed to be secure if the adversary has very little chance to compute certain properties of the system. Threat models for four major scenarios are:

1. For a cryptographically-enforced access control method using encryption, the adversary should not be able to identify the corresponding plaintext data by looking at the ciphertext, even if given the choice of a correct and an incorrect plaintext. This should hold true even if parties excluded by the access control policy collude among each other and with the adversary.

2. For a cryptographic protocol for searching and filtering encrypted data, the adversary should not be able to learn anything about the encrypted data beyond whether the corresponding predicate was satisfied. Recent research has also succeeded in hiding the search predicate itself so that a malicious entity learns nothing meaningful about the plaintext or the filtering criteria.
3. For a cryptographic protocol for computation on encrypted data, the adversary should not be able to identify the corresponding plaintext data by looking at the ciphertext, even if given the choice of a correct and an incorrect plaintext. Note that this is a very stringent requirement because the adversary is able to compute the encryption of arbitrary functions of the encryption of the original data. In fact, a stronger threat model called chosen ciphertext security for regular encryption does not have a meaningful counterpart in this context – the search to find such a model continues [38].
4. For a cryptographic protocol ensuring the integrity of data coming from an identified source, there could be a range of threat models. The core requirement is that the adversary should not be able to forge data that did not come from the purported source. There could also be some degree of anonymity in the sense that the source could only be identified as being part of a group. In addition, in certain situations (maybe legal), a trusted third party should be able to link the data to the exact source.

7.3 Analysis

Given a threat model, a candidate cryptographic protocol is proved to defend against the threat by either a reduction argument or a simulation argument. A reduction argument proves that if an adversary can compute certain properties of the system, then it can break a number of theoretical properties widely assumed to be hard. Alternatively, it can break the security of simpler cryptographic primitives that were used as building blocks of the system. In a simulation argument, such as that referenced in Canetti [39], it is proved that if the adversary can break a candidate system, then a "simulator" can break an ideal functionality that naturally represents the security of the system. We list some of the current research in the abovementioned areas that are proved to be secure in their respective models.

1. Identity and attribute based encryption [49], [50] methods enforce access control using cryptography. In identity-based systems, plaintext can be encrypted for a given identity and the expectation is that only an entity with that identity can decrypt the ciphertext. Any other entity will be unable to decipher the plaintext, even with collusion. Attribute-based encryption extends this concept to attribute-based access control.
2. Boneh and Waters [40] construct a public key system that supports comparison queries, subset queries and arbitrary conjunction of such queries.
3. In a breakthrough result [41] in 2009, Gentry constructed the first fully homomorphic encryption scheme. Such a scheme allows one to compute the encryption of arbitrary functions of the underlying plaintext. Earlier results [42] constructed partially homomorphic encryption schemes.
4. Group signatures [43] enable individual entities to sign their data but remain identifiable only in a group to the public. Only a trusted third party can pinpoint the identity of the individual.

7.4 Implementation

The current algorithms to implement identity/attribute-based encryption schemes and group signatures use elliptic curve groups that support bilinear pairing maps. This makes group elements somewhat larger to represent. Additionally, the pairing operations are computationally expensive.

Gentry's original construction of a fully homomorphic encryption (FHE) scheme used ideal lattices over a polynomial ring. Although lattice constructions are not terribly inefficient, the computational overhead for FHE is still far from practical. Research is ongoing to find simpler constructions [44], [45], efficiency improvements [46], [47] and partially homomorphic schemes [48] that suffice for an interesting class of functions.

8.0 Granular Access Control

The security property that matters from the perspective of access control is secrecy – preventing access to data by people that should not have access. The problem with course-grained access mechanisms is that data that could otherwise be shared is often swept into a more restrictive category to guarantee sound security. Granular access control gives data managers more precision when sharing data, without compromising secrecy.

8.1 Use Case

Big Data analysis and cloud computing are increasingly focused on handling diverse data sets, both in terms of variety of schemas and of security requirements. Legal and policy restrictions on data come from numerous sources. The Sarbanes-Oxley Act levees requirements to protect corporate financial information, and HIPAA includes numerous restrictions on sharing personal health records. Executive order 13526 outlines an elaborate system of protecting national security information. Privacy policies, sharing agreements, and corporate policy also impose requirements on data handling.

Managing this plethora of restrictions has resulted in increased costs for developing applications and a walled garden approach in which few people can participate in the analysis. Granular access control is necessary for analytical systems to adapt to this increasingly complex security environment.

8.2 Modeling

The primary threat vector related to granular access control comes from its inconsistent incorporation into the application layer. Getting granular access correct requires a rigorous approach. Not only does this make application development expensive and complicated, but the variety of applications introduces many opportunities to get granular access controls wrong.

Granular access control can be decomposed into three sub-problems. The first is keeping track of secrecy requirements for individual data elements. In a shared environment with many different applications, the ingesters (those applications that contribute data) need to communicate those requirements to the queriers. This coordination requirement complicates application development, and is often distributed among multiple development teams.

An additional challenge in tracking these requirements is to maintain access labels across analytical transformations. An element created from two or more other elements may be restricted at the least upper bound of the restrictions of the other elements, according to some lattice. However, some data access requirements don not follow a lattice, such as aggregated medical records that could be broadly releasable, while the elements that contribute to those aggregates are highly restricted. Tracking these requirements is also implemented in the application space, and can be difficult to get right.

The second sub-problem is keeping track of roles and authorities for users. Once a user is properly authenticated, it is still necessary to pull security-related attributes for that user from one or more trusted sources. LDAP, Active Directory, OAuth, OpenID, and many other systems have started to mature in this space. One of the continual challenges is to properly federate the authorization space, so a single analytical system can respect roles and authorities that are defined across a broad ecosystem.

The third sub-problem is properly implementing secrecy requirements with mandatory access control. This is a logical filter that incorporates the requirements coupled with the data and the attributes coupled with a user to make an access decision. This filter is usually implemented in the application space because there are few infrastructure components that support the appropriate level of granular access controls.

8.3 Analysis

The primary threat vector associated with granular access control is the reliance on implementation in the application space. A natural approach to mitigating this threat is to reduce the complexity of the addition of granular access controls to an application. In other words, implement as much as possible in the infrastructure layer, and adapt standards and practices to simplify those concerns that still exist in the application space.

The first challenge is to pick the appropriate level of granularity required for a given domain. Row-level protection, where a row represents a single record, is often associated with security that varies by data source. Column-level protection, where a column represents a specific field across all records, is often associated with sensitive schema elements. A combination of row- and column-level security is more granular still but can break down under analytical uses. Table transformations, or question-focused data sets, often do not preserve row- or column-orientation, so they require a solution with even finer granularity. Cell-level access control supports labeling every atomic nugget of information with its releasability and can support a broad set of data transformation uses.

Unfortunately, narrowing the aperture of access labels is difficult. Several scalability challenges arise when scaling solutions up to petabyte-sized data sets with numerous contributors on a globally-distributed network. These challenges include label storage, query-time access checks, and update costs as authorities and access restrictions change over time.

To keep costs low, it is important to model the expectation of change over time. For example, there are relatively immutable attributes of log data, such as where the log was generated and what system generated it. There are also highly mutable attributes associated with users, such as current employment or current training. When modeling the cost of reads, writes, and system maintenance, one would prefer a solution in which the mutable elements are normalized to reduce the cost of updates, while the immutable elements are denormalized to improve query performance. This creates an approach of explicitly storing many of the access control requirements alongside the granular data elements, and dynamically joining the remaining attributes at query time.

8.4 Implementation

Implementing granular security access requires elements that span the Big Data ecosystem. Protocols for tracking access restrictions alongside data are needed and should be implemented in storage systems, such as HDFS and NoSQL databases. Many of the other challenges described in this paper are prerequisites for a full granular access control solution, including authentication and mandatory access control.

Apache Accumulo is one example of a NoSQL database that supports mature, cell-level access control. In Accumulo, every atomic key/value pair is tagged with an expression that describes the roles required to read that entry, and every query includes a role check. For many uses, the compression and caching techniques in Accumulo have a negligible impact on performance and greatly simplify those elements of fine-grained access control that must be implemented in the application.

9.0 Granular Audits

With real-time security monitoring (see Section 5), notification at the moment an attack takes place is the goal. In reality, this will not always be the case (e.g., new attacks, missed true positives). In order to discover a missed attack, audit information is necessary. Audit information is crucial to understand what happened and what went wrong. It is also necessary due to compliance, regulation and forensic investigation. Auditing is not something new, but the scope and granularity might be different in real-time security contexts. For example, in these contexts there are more data objects, which are probably (but not necessarily) distributed.

9.1 Use Case

Compliance requirements (e.g., PCI, Sarbanes-Oxley) require financial firms to provide granular auditing records. Additionally, the cost of losing records containing private information is estimated at $200/record. Legal action – depending on the geographic region – might follow in case of a data breach. Key personnel at financial institutions require access to large data sets containing personally identifiable information (PII), such as social security numbers. In another potential use case, marketing firms want access to personal social media information to optimize their deployment of online ads.

9.2 Modeling

Key factors for auditing comprise the following:

1. Completeness of the required audit information (i.e., having access to all the necessary log information from a device or system).
2. Timely access to audit information. This is especially important in case of forensics, for example, where time is of the essence.
3. Integrity of the information or, in other words, audit information that has not been tampered with.
4. Authorized access to the audit information. Only authorized people can access the information and only the parts they need to perform their job.

Threats (e.g., unauthorized access, removal of data, tempering with log files) to those key factors will jeopardize the audit data and process.

9.3 Analysis

Auditing capabilities need to be enabled across the Big Data infrastructure. The specific capabilities depend on the supported auditing features of the infrastructure components. Examples are log information from network components (e.g., router syslog), applications, operating systems, and databases. The challenge is to create a cohesive audit view of an attack using the available audit information of the different components (but not all components may be able to deliver the necessary information).

9.4 Implementation

Implementation of audit features starts on the individual component level. Examples include enabling syslog on routers, application logging, and enabling logging on the operating system level. After this, a forensics or SIEM tool collects, analyzes and processes this information. The amount of recording is subject to the limitations of the SIEM tool in processing the volume and velocity of the audit data. Ironically, the audit data might have the characteristics of Big Data itself and, as such, may need to be processed by a Big Data infrastructure. To separate the use of the Big Data infrastructure and the audit of this infrastructure, it is recommended to have the forensics/SIEM tool implemented and used outside of the Big Data infrastructure when feasible. Another approach would be to create an "Audit Layer/Orchestrator," which would abstract the required (technical) audit information from the auditor. This orchestrator would take the auditor's requests (i.e., who had access to data object X on date D), collect the necessary audit information from the required infrastructure components, and returns this information to the auditor.

10.0 Data Provenance

Provenance metadata will grow in complexity due to large provenance graphs generated from provenance-enabled programming environments in Big Data applications. Analysis of such large provenance graphs to detect metadata dependencies for security and/or confidentiality applications is computationally intensive.

10.1 Use Case

Several key security applications require a digital record with, for example, details about its creation. Examples include detecting insider trading for financial companies or determining the accuracy of the data source for research investigations. These security assessments are time-sensitive in nature and require fast algorithms to handle the provenance metadata containing this information. In addition, data provenance complements audit logs for compliance requirements, such as PCI or Sarbanes-Oxley.

10.2 Modeling

Secure provenance in Big Data applications first requires the provenance records to be reliable, provenance-integrated, privacy-preserving, and access-controllable. At the same time, due to the characteristics of Big Data, provenance availability and scalability should also be carefully addressed. Specifically, the threats to the provenance metadata in Big Data applications can be formally modeled into three categories:

1. Malfunctioning Infrastructure Components – In Big Data applications, when large numbers of components collaboratively generate large provenance graphs from provenance-enabled programming environments, it is inevitable that some infrastructure components could sporadically malfunction. Once the malfunction occurs, provenance data cannot be timely generated and some malfunctions could lead to incorrect provenance records. As a result, the malfunctioning infrastructure components will reduce the provenance availability and reliability.
2. Infrastructure Outside Attacks – Since provenance is pivotal to the usability of Big Data applications, it naturally becomes a target in Big Data applications. An outside attacker can forge, modify, replay, or unduly delay the provenance records during its transmission to destroy the usability of the provenance, or violate privacy by eavesdropping and analyzing the records.
3. Infrastructure Inside Attacks – Compared to the outside attacks, the infrastructure inside attacks are more harmful. An inside attacker could modify and delete the stored provenance records and audit logs to destroy the provenance system in Big Data applications.

10.3 Analysis

To address the above threats, two research issues need to be engaged to ensure the trustworthiness and usability of secure provenance in Big Data applications (i.e., securing provenance collection and fine-grained access control of provenance).

To secure provenance collection, the source components that generate provenance in the infrastructure should be first authenticated. In addition, periodic status updates should be generated to ensure the health of the source components. To guarantee the accuracy of the provenance records, the provenance should be taken through an integrity check to assure that it is not forged or modified. Furthermore, the consistency between the provenance and its data should also be verified because inconsistency can lead to wrong decisions. Since the provenance sometimes contains sensitive information pertaining to the data, encryption techniques are required to achieve provenance confidentiality and privacy preservation [32]. Finally, compared to the small-scale, static data applications, the provenance collection for Big Data should be efficient to accommodate ever-increasing volume, variety, and velocity of information. In this way, secure provenance collection is efficiently achieved in Big Data applications. In other words, the provenance collection is secure against malfunctioning infrastructure components and outside attacks.

To resist the infrastructure inside attacks, fine-grained access control of provenance is desired. In Big Data applications, the provenance records include not only the provenance of different application data, but also the provenance for the Big Data infrastructure itself. Therefore, the number of provenance records in Big Data applications is much larger than that in small-scale, static data applications. For these large-volume, complex, and sometimes sensitive provenance, access control is required. Otherwise, it would not be possible to deal with the infrastructure inside attacks. Fine-grained access control assigns different rights to different roles to access provenance in Big Data applications. At the same time, data independent persistence should also be satisfied when updating the large provenance graphs. For example, even though a data objective is removed, since it serves as an ancestor of other data objectives, its provenance should be kept in the provenance graph. Otherwise, the provenance graph will become disconnected and incomplete. Furthermore, fine-grained access control should be dynamic and scalable, and flexible revocation mechanisms should also be supported.

10.4 Implementation

Provenance is crucial for the verification, audit trails, assurance of reproducibility, trust, and fault detection in many big-data applications. To retrofit provenance in existing cloud infrastructure for Big Data applications, we must address secure provenance collection and fine-grained access control effectively. To address the secure provenance collection, fast and lightweight authentication technique should be integrated into the current provenance in existing cloud infrastructure (e.g., PASOA [33]). In addition, secure channels should be established between infrastructure components to achieve end-to-end security. Finally, fine-grained access control (e.g., revocable ABE access control [34]) should be integrated into the current provenance storage system (e.g., PASS [35]) to achieve provenance storage and access security in Big Data applications.

Conclusion

Big Data is here to stay. It is practically impossible to imagine the next application without it consuming data, producing new forms of data, and containing data-driven algorithms. As computing environments become cheaper, application environments become networked, and system and analytics environments become shared over the cloud, security, access control, compression, encryption and compliance introduce challenges that must be addressed in a systematic way.

In this paper, we have highlighted the top ten security and privacy problems that need to be addressed to make Big Data processing and computing infrastructure more secure. Common elements specific to Big Data arise from the use of multiple infrastructure tiers (both storage and computing) for processing Big Data; the use of new compute infrastructures such as NoSQL databases (for fast throughput necessitated by Big Data volumes) that have not been thoroughly vetted for security issues; the non-scalability of encryption for large data sets; the non-scalability of real-time monitoring techniques that might be practical for smaller volumes of data; the heterogeneity of devices that produce the data; and the confusion surrounding the diverse legal and policy restrictions that lead to ad hoc approaches for ensuring security and privacy. Many of the items in this list serve to clarify specific aspects of the attack surface of the entire Big Data processing infrastructure that should be analyzed for these threats.

Our hope is that this paper will spur action in the research and development community to collaboratively focus on the barriers to greater security and privacy in Big Data platforms.

References

[1] I. Roy, S. T. V. Setty, A. Kilzer, V. Shmatikov and E. Witchel, "Airavat: security and privacy for MapReduce" in USENIX conference on Networked systems design and implementation, pp 20-20, 2010.

[2] L. Okman, N. Gal-Oz, Y Gonen, E. Gudes and J Abramov, "Security Issues in NoSQL Databases" in TrustCom IEEE Conference on International Conference on Trust, Security and Privacy in Computing and Communications, pp 541-547, 2011.

[3] Srini Penchikala, "Virtual Panel: Security Considerations in Accessing NoSQL Databases", Nov. 2011. http://www.infoq.com/articles/nosql-data-security-virtual-panel.

[4] B. Sullivan, "NoSQL, But Even Less Security", 2011. http://blogs.adobe.com/asset/files/2011/04/NoSQL-But-Even-Less-Security.pdf.

[5] E. Chickowski, "Does NoSQL Mean No Security?", Jan. 2012, http://www.darkreading.com/database-security/167901020/security/news/232400214/does-nosql-mean-no-security.html.

[6] K. Ding and W. Huang, "NoSQL, No Injection!?" in Def Con 18 Hacking Conference, Jul. - Aug. 2010.

[7] N. Tippenhauer, C. Pöpper, K. Rasmussen, S. Capkun, "On the requirements for successful GPS spoofing attacks," in Proceedings of the 18th ACM conference on Computer and communications security, pp. 75-86, Chicago, IL, 2011.

[8] P. Gilbert, J. Jung, K. Lee, H. Qin, D. Sharkey, A. Sheth, L. P. Cox, "YouProve: Authenticity and Fidelity in Mobile Sensing," ACM SenSys 2011, Seattle, WA, November, 2011.

[9] B. Levine, C. Shields, N. Margolin, "A Survey of Solutions to the Sybil Attack," Tech report 2006-052, University of Massachusetts Amherst, Amherst, MA, October 2006.

[10] B. Agreiter, M. Hafner, and R. Breu, "A Fair Non-repudiation Service in a Web Service Peer-to-Peer Environment," Computer Standards & Interfaces, vol 30, no 6, pp. 372-378, August 2008.

[11] A. Anagnostopoulos, M. T. Goodrich, and R. Tamassia, "Persistent Authenticated Dictionaries and Their Applications," in Proceedings of the 4th International Conference on Information Security, pp. 379-393, October, 2001.

[12] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proceedings of the 4th international conference on Security and privacy in communication networks (SecureComm '08), pages 9:1–9:10, New York, NY, USA, 2008.

[13] W. Bagga and R. Molva, "Collusion-Free Policy-Based Encryption," ISC, volume 4176 of Lecture Notes in Computer Science, pp 233-245. Springer, 2006.

[14] D. Boneh, C. Gentry, and B. Waters, "Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys," Lecture Notes in Computer Science, 2005.

[15] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in Proceedings of the 16th ACM conference on Computer and communications security (CCS '09), pages 213–222, New York, NY, USA, 2009.

[16] J. Feng, Y. Chen, and D. Summerville, "A Fair Multi-Party Non-Repudiation Scheme for Storage Clouds," the 2011 International Conference on Collaboration Technologies and Systems (CTS 2011), Philadelphia, PA., USA, May 23 - 27, 2011.

[17] J. Feng, Y. Chen, W.-S. Ku, and P. Liu, "Analysis of Integrity Vulnerabilities and a Non-repudiation Protocol for Cloud Data Storage Platforms," the 2nd International Workshop on Security in Cloud Computing (SCC 2010), in conjunction with ICPP 2010, San Diego, California, USA, Sept. 14, 2010.

[18] J. Feng, Y. Chen, and P. Liu, "Bridging the Missing Link of Cloud Data Storage Security in AWS," the 7th IEEE Consumer Communications and Networking Conference - Security for CE Communications (CCNC '10), Las Vegas, Nevada, USA, January 9 - 12, 2010.

[19] A. Juels and B. S. K. Pors Jr., "Proofs of retrievability for large files," in Proc. ACM CCS, pages 584–597, 2007.

[20] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable Secure File Sharing on Untrusted Storage," in USENIX Conference on File and Storage Technologies (FAST), pages 29-42, 2003.

[21] M. Majuntke, D. Dobre, M. Serafini, and N. Suri, "Abortable Fork-Linearizable Storage," in Proc. of OPODIS, p255-269, 2009.

[22] K. K. Muniswamy-Reddy, P. Macko, and M. Seltzer, "Provenance for the Cloud," the 8th USENIX Conference on File and Storage Technologies (FAST '10), Feb. 2010.

[23] J. Onieva, J. Lopez, and J. Zhou, "Secure Multi-Party Non-repudiation Protocols and Applications," Advances in Information Security Series, Springer, 2009.

[24] R. A. Popa, J. Lorch, D. Molnar, H. J. Wang, and L. Zhuang, "Enabling Security in Cloud Storage SLAs with CloudProof," Microsoft TechReport MSR-TR-2010-46, May, 2010.

[25] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in INFOCOM, 2010 Proceedings IEEE, pages 1 –9, march 2010.

[26] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," Parallel and Distributed Systems, IEEE Transactions on, 22(5):847 –859, May 2011.

[27] D. Boyd, and K. Crawford. "Criticial Questions for Big Data," in Information, Communication & Society, 15:5, pp. 662-675, May 10, 2012.

[28] C. Duhigg. "How Companies Learn Your Secrets," The New York Times, February 16, 2012.

[29] M. Barbard and T. Zeller, "A Face is Exposed for AOL Searcher No. 4417749". The New York Times, August 9, 2006.

[30] A. Hough. "Google engineer fired for privacy breach after 'staking and harassing teenagers'". The Telegraph. Sept 15, 2010.

[31] Electronic Privacy Information Center. "Re-identification," http://epic.org/privacy/reidentification. Retrieved Nov 1, 2012.

[32] R. Lu, X. Lin, X. Liang, and X. Shen, "Secure provenance: the essential of bread and butter of data forensics in cloud computing", in Proceeding of 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS '10) pp. 282--292, New York, NY, USA, 2010.

[33] Provenance Aware Service Oriented Architecture (PASOA) Project, http://www.pasoa.org/.

[34] V. Goyal, O. Pandey, A. Sahai, B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data", ACM Conference on Computer and Communications Security, pp. 89-98, 2006.

[35] K. Muniswamy-Reddy, D. Holland, U. Braun, and M. Seltzer, "Provenance-aware storage systems", in USENIX Annual Technical Conference, General Track, USENIX, 2006, pp. 43-56.

[36] C. Percival, "Cache missing for fun and profit", BSDCan, 2005.

[37] Acıiçmez, Onur, Çetin Koç, and Jean-Pierre Seifert. "Predicting secret keys via branch prediction." Topics in Cryptology–CT-RSA 2007 (2006): 225-242.

[38] J. Loftus and A. May and N.P. Smart and F. Vercauteren, "On CCA-Secure Fully Homomorphic Encryption", Cryptology ePrint Archive, Report 2010/560, 2010, http://eprint.iacr.org.

[39] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols." Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on. IEEE, 2001.

[40] D. Boneh, and B. Waters. "Conjunctive, subset, and range queries on encrypted data." Theory of Cryptography (2007): 535-554.

[41] C. Gentry. "Fully homomorphic encryption using ideal lattices." Proceedings of the 41st annual ACM symposium on Symposium on theory of computing-STOC'09. ACM Press, 2009.

[42] D. Boneh, E.-J. Goh, and K. Nissim. "Evaluating 2-DNF formulas on ciphertexts." Theory of Cryptography (2005): 325-341.

[43] D. Boneh, X. Boyen and H. Shacham. "Short group signatures." Advances in Cryptology–CRYPTO 2004. Springer Berlin/Heidelberg, 2004.

[44] M. van Dijk, C. Gentry, S. Halevi and V. Vaikuntanathan, "Fully homomorphic encryption over the integers". Advances in Cryptology–EUROCRYPT 2010, 24-43.

[45] J.S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, “Fully homomorphic encryption over the integers with shorter public keys”. Advances in Cryptology–CRYPTO 2011, 487-504.

[46] C. Gentry, S. Halevi, N.P. Smart: Homomorphic Evaluation of the AES Circuit. CRYPTO 2012: 850-867

[47] Craig Gentry, Shai Halevi, Nigel P. Smart: Fully Homomorphic Encryption with Polylog Overhead. EUROCRYPT 2012: 465-482.

[48] M. Naehrig, K. Lauter and V. Vaikuntanathan. “Can homomorphic encryption be practical?.” Proceedings of the 3rd ACM workshop on Cloud computing security workshop. ACM, 2011.

[49] D. Boneh and M. Franklin. “Identity-based encryption from the Weil pairing.” SIAM Journal on Computing 32.3 (2003): 586-615.

[50] V. Goyal, O. Pandey, A. Sahai, and B. Waters (2006, October). Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM conference on Computer and communications security (pp. 89-98). ACM.

[51] T. Ptacek and T. Newsham. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Tech. Report. 1998.

[52] M. Barreno, B. Nelson, R. Sears, A. Joseph, J.D. Tygar. “Can Machine Learning be Secure?”. Proc. Of the 2006 ACM Symposium on Information, Computer, and Communications Security, ASIACCS 2006, pp. 16-25, March 21-24, 2006.

[53] W. Diffie, S. Landau. Privacy on the Line: The Politics of Wiretapping and Encryption. The MIT Press, 2010.

[54] C. Gentry, “Computing arbitrary functions of encrypted data,” Communications of the ACM, Vol. 53, No. 3, 2010.

[55] S. Kamara and K. Lauter, “Cryptographic cloud storage,” Financial Cryptography and Data Security, 2010.

[56] http://www.emc.com/leadership/digital-universe/iview/big-data-2020.htm

