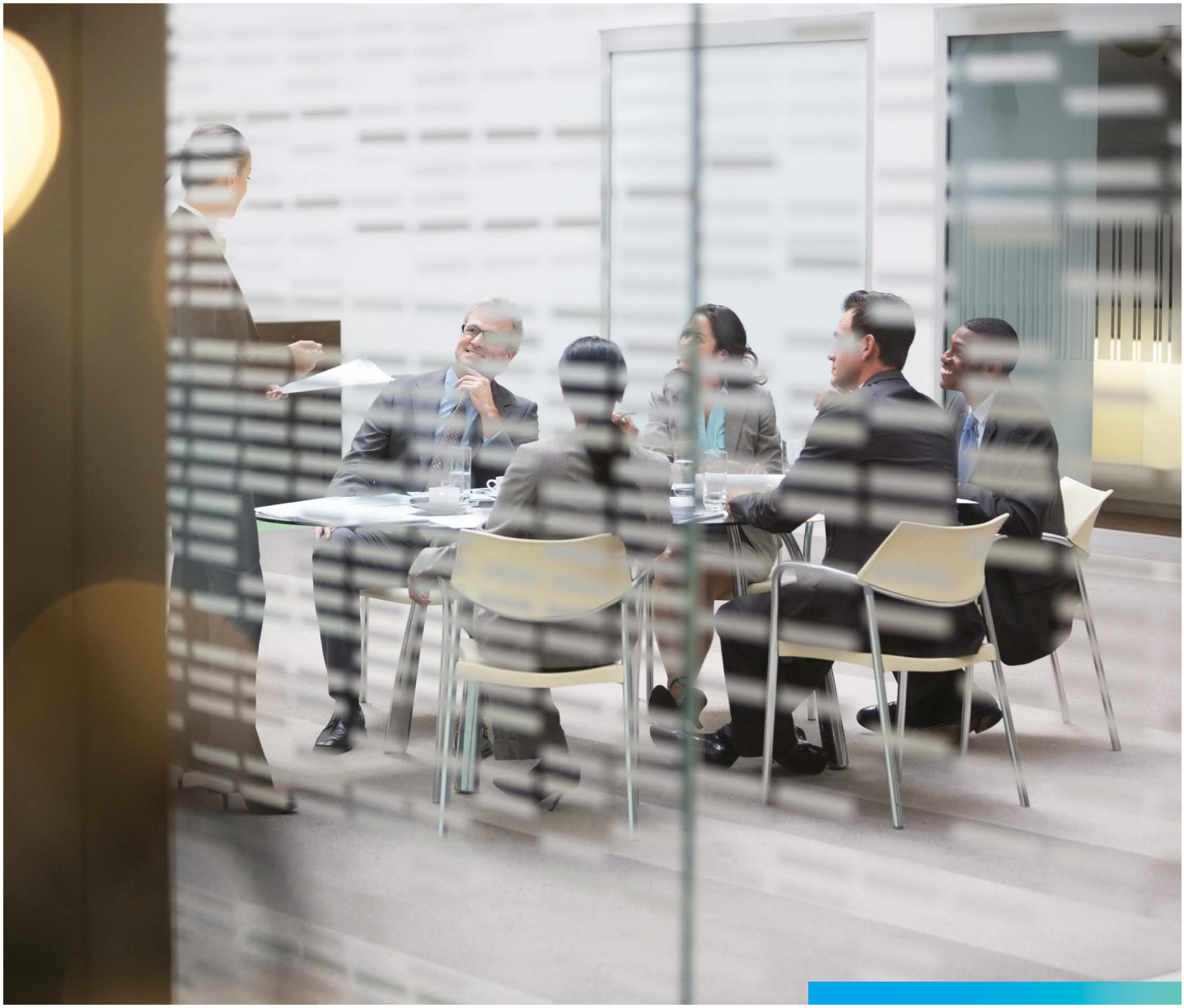

Brochure

Security

Protecting Enterprise Data in Hadoop

Voltage SecureData for Hadoop

MICRO FOCUS

Introduction

Big Data is an exciting concept and emerging set of technologies that hold seemingly unlimited promise to enable organizations to gain new analytic insights and operational efficiencies. It is a unique architecture that enables low-cost, high-speed, parallel processing of huge data sets of structured and unstructured data. In the 2017 IDG Enterprise Security Priorities Survey, the largest segment of respondents at 42%, identified Big Data Analytics as one of the top two categories for new or increased spending in the next year.*

This Big Data movement is supported by multiple, open source initiatives centered on Apache Hadoop. The core driving forces have been providing flexibility, performance and scalability through the use of multiple standard, low-cost processing nodes and the Hadoop Distributed File System (HDFS) on commodity off-the-shelf hardware. Security was not a key design criterion.

When used in an enterprise environment, however, the importance of data security becomes paramount. Organizations must protect sensitive customer, partner and internal information from an increasing array of advanced threats and risks, and must adhere to a complex set of privacy laws and compliance requirements. But, by its nature, Hadoop poses many unique challenges to properly securing this environment, including:

- Inability to guarantee complete removal of sensitive data once it has entered the cluster (as Hadoop does not guarantee when a file is actually removed from the Trash plus there are no secure wipe within HDFS to ensure all data blocks have been purged).
- Rapidly evolving tools with frequent releases, driven by a diverse and well funded open- source developer community.
- Lack of a core set of security controls that are commonplace on commercial, relational database management system (RDBMS) products.
- Multiple sources of data coming from multiple enterprise systems and real-time feeds with varying (or unknown) protection requirements.
- Multiple types of data (structured RDBMS- based, file-based, text feeds, etc.) combined together in the Hadoop "data lake."
- Access by many different users with varying analytic needs and ad-hoc processes, with the likelihood of propagating to other enterprise systems and tools.
- Automatic and largely uncontrollable replication of data across multiple nodes once entered into the HDFS data store.
- Reduced physical and logical access controls if Hadoop clusters are deployed remotely or in a cloud environment.

Further exacerbating these risks is the fact that the aggregation of data found within these Big Data systems makes for an extremely alluring target for hackers and data thieves. Hadoop presents brand new challenges to data risk management: the potential concentration of vast amounts of sensitive corporate and personal data in a low-trust environment. The data-concentration issue alone can lead to far-reaching business risks through the exposure of a company's data assets if data access, analysis, and extraction are not strictly controlled. A bank's portfolio of positions and transactions may be visible and searchable. An insurance company's entire risk posture may be easily determined. A government's military health information may be indirectly exposed from health analytics. Fully securing Hadoop in enterprise environments is essential to mitigate these potentially huge Big Data exposures.

Basic Hadoop Security Controls

There are a number of traditional IT security controls that should be put in place as the basis for securing a Hadoop environment. All the standard perimeter protection controls apply—network firewalls, intrusion detection and protection systems (IDPS), security information and event management (SIEM), configuration control, vulnerability management, etc. These are the base level of general security controls.

For the next level of security, the open source community has been investing heavily in building Hadoop-specific tools and best practices

2017 IDG Enterprise Security Priorities Survey, page 19.

to provide enterprise grade security. There are four key areas of focus: authentication, authorization, audit and data protection.

Authentication is fundamental to ensuring users are who they claim to be. It can be best accomplished by integrating with existing enterprise identity and access management services. Apache Knox provides a REST API Gateway as a single point of access to the Hadoop cluster (or multiple clusters). Using Kerberos provides strong authentication to establish identities for clients, hosts and services. These tools can be integrated with an organization's LDAP server, including Microsoft Active Directory.

Much of the issue around Hadoop authorization has been the limited ability to grant or deny access privileges on a granular basis. This is improving, as SQL-style authorization to Hive and HDFS ACLs are now available in Hadoop. The capability to audit and track individual user accesses to specific services and HDFS data components is also improving with the use of the above tools.

The fourth pillar of Hadoop security is data protection. Even with all the above controls, it is a given that at some point unauthorized users will gain inappropriate access to Hadoop data. This is why protecting the data itself through encryption or other de-identification techniques is of paramount importance.

De-identified data in Hadoop is protected data, and even in the event of a data breach, yields nothing of value, avoiding the penalties and costs such an event would otherwise have triggered.

Data Protection Methodologies

There are multiple approaches that can be deployed to directly protect the data in a Hadoop environment:

Volume Storage and File-Level Encryption

This is a fairly standard type of encryption frequently deployed by IT to protect sensitive data in file systems. This is often referred to as "data-at-rest" encryption. The data is encrypted at the file system or disk volume level, and is protected while residing "at rest" on the data store. This protects against unauthorized personnel who may have physically obtained the disk from being able to read anything from it. This is a useful control in a large Hadoop cluster due to frequent disk repairs and swap outs.

Volume-level encryption using base-level Linux OS capabilities has been available for some time, and there is considerable effort right now to add HDFS-level encryption to Hadoop, which would provide more granular control at the file system level.

However, both these approaches do nothing to protect the data from any and all access when the disk is running within the system—which, of course, is essentially all the time. Decryption is applied automatically when the data is read by the operating system, and live, vulnerable data is fully exposed to any user or process accessing the system, whether authorized or not. Further, every read and every write invokes encryption and decryption, which adds overhead, even if the data being written or read is not sensitive.

Transport-Level Encryption

This is another common data encryption technique used to protect data while being transferred across a network, and is sometimes called "data-in-motion" or "wire" encryption. SSL/TLS protocols are a common example of this. Wire encryption is currently receiving a lot of attention from the Hadoop community to ensure Hadoop programs can support these secure transport-level protocols. This is an important data protection technique, as data moving "across the wire" is often most vulnerable to interception and theft, whether transferring data across clusters in a rack within the datacenter, nodes in the cloud, or especially across a public network (i.e., the Internet).

However, the protection this method affords is limited to the actual transfer itself. Clear, unprotected data exists at the source, and clear, unprotected data is automatically reassembled at the destination. This provides many vulnerable points of access for the potential thief, and like Storage-level encryption above, does nothing to protect the data from unauthorized users who have obtained fraudulent access to the system itself.

Data Masking

This is a useful technique for obfuscating sensitive data, commonly used for creation of test and development data or analytic data from live production information. A typical system will support multiple transformation techniques, from extremely simple (replacing digits with "X's") to quite complex (replacing valid addresses with different valid addresses from the same area). Depending on the requirements and how cleverly you select the transformations, masked data can sometimes be used in its protected state for analytic applications. However, there are several limitations of data masking in the Hadoop environment:

- Masked data is intended to be irreversible, which limits its value for many analytic applications and post-processing requirements. For example, a common Hadoop use case is taking data from live systems, performing low cost analysis in Hadoop, and then pushing

downstream processing to existing Business Intelligence (BI) tools where live data is again needed for business action and decision.

- There is no guarantee the specific masking transformation chosen for a specific sensitive data field fully obfuscates it from identification, particularly when analyzed with adjacent, unaltered fields or correlated with other data. For example, in a recently released dataset of supposedly masked information on 173 million taxi rides in New York City, individual taxis and their drivers could be re-identified using related data because of a weak hashing algorithm used to alter the license plate numbers.
- Many data masking transformations create duplicates and destroy referential integrity, resulting in join operations on data base tables not mapping properly and reducing the ability to perform analytic functions on the data.
- Specific masking techniques may or may not be accepted by auditors and assessors, affecting whether they truly meet compliance requirements and provide safe harbor in the event of a breach.
- The masking process, depending on the transformation, may require mapping tables, making processing quite slow and not scalable. These mapping tables create another place to protect sensitive data, which at Hadoop scale, may be very expensive or even impossible to achieve.

Data-Centric Security

A data-centric security approach is quite different than the above techniques, and calls for de-identifying the data as close to its source as possible, replacing the sensitive data elements with usable, yet de-identified, equivalents that retain their format, behavior and meaning. This is also called "end-to-end data protection" and provides an enterprise-wide solution for data protection that extends beyond the Hadoop environment.

This protected form of the data can then be used in subsequent applications, analytic engines, data transfers and data stores. For Hadoop, the best practice is to never allow sensitive information to reach the HDFS in its live and vulnerable form.

Micro Focus Data-Centric Security

To properly protect data in Hadoop, you need to employ a data-centric security approach. As the above examples illustrate, this cannot be achieved by simply deploying piecemeal encryption or data masking within Hadoop. It requires protecting data-at-rest, in-use, and in-motion; as close to its source as possible, and doing so in way that is scalable, format preserving and maintains the analytic meaning and logic of the data without live data exposure risk. It requires protecting the data at the field level in a way that it can be used by applications in its de-identified state, while also being selectively and securely re-identified for those specific applications and users that require it. Micro Focus® Voltage SecureData provides such a solution.

Three core technology breakthroughs enable SecureData to meet these requirements:

Format-Preserving Encryption

A fundamental innovation enabling SecureData data-centric platform is Micro Focus Voltage Format-Preserving Encryption (FPE), providing high strength encryption of data without altering the original data format and preserving business value and referential integrity across distributed data sets. This enables applications, analytic processes and databases to use the protected data without alteration, even across distributed systems, platforms and tools. Protection is applied at the field or even partial-field level, leaving non-sensitive portions of fields available for applications while protecting the sensitive parts. FPE preserves referential integrity, which means protected data can still be consistently referenced and joined across tables and data sets—a major requirement for proper operations with the mix of data entered into Hadoop, and especially critical where common identifiers like Social Security Numbers or ID's are used as common references across disparate data sets.

Policy controlled secure reversibility enables data to be selectively re-identified in trusted systems which need live data, enabling full end-to-end data processes to be secured without resorting to risky and cumbersome mapping databases. Format-Preserving Encryption has also received strong attention from the government, and is recognized as a NIST-standard mode of standard AES encryption, specifically FF1 mode AES defined in NIST 800-38G. This provides users confidence in the security proofs and standards underpinning FPE. SecureData has built a robust eco-system around FPE, providing support across multiple enterprise platforms with proven implementation tools, delivering 'always-on' data protection.

Secure Stateless Tokenization

Augmenting the FPE encryption technology is Micro Focus Voltage Secure Stateless Tokenization (SST).

Tokenization is the process of replacing live data with a random surrogate. Traditional approaches use databases to map live to surrogate values, which limit performance and scale, rendering it impractical for most Hadoop use cases. SST technology is stateless, eliminating the need for a token database. Instead, the system uses a pre-generated token mapping table containing random numbers using a proven,

independently validated random token mapping method. This eliminates the complexities of token database synchronization, back-up and recovery, and provides a 100% guarantee of data integrity with no data collisions. Tokenization is an important de-identification capability for credit card numbers, and proper use of this feature can help ensure your Hadoop environment stays out of scope for PCI DSS compliance requirements, avoiding cost and disruptive audit processes.

Combining FPE with SST gives the user the necessary flexibility to select the best data protection technique for each data type and regulatory mandate, all within a single solution. Both FPE and SST permit data to be de-identified at scale. De identification tasks can be processed in parallel across multiple nodes, enabling massive scale and performance without additional infrastructure or complexity.

Stateless Key Management

Micro Focus Voltage Stateless Key Management provides keys to be derived as needed with no storage or database management issues because database synchronization and frequent backups are not required. Because keys are dynamically generated, there is no need for key backup, and no possibility of key loss. Key management can be linked to existing identity management infrastructure, including external LDAP directories. Permission to decrypt or de-tokenize can be assigned on an application policy basis, and can incorporate user roles and groups to simplify management based on identity management system policy. This role based access to data at the field level is extremely important in the Hadoop environment for enabling applications and users to decrypt exactly the data they are authorized to access, presenting different views of clear and encrypted fields to match specific user permissions. Stateless Key Management also enables simplified implementation and provides high-performance, scalable, distributed processing that is well-matched with the Hadoop architecture.

SecureData Deployment Architecture

Implementing data-centric security involves installing the SecureData infrastructure components and then interfacing with the appropriate applications and data flows. Infrastructure deployment is simplified by the use of a SecureData virtual software appliance that includes the Key Server, Management Console, Web Services Server, etc. Developer templates, APIs and command line tools enable encryption and tokenization to occur natively on the widest variety of platforms, including Linux, mainframe and mid-range computers, and support integration with a broad range of infrastructure components, including ETL, databases, and, of course, programs running in the Hadoop environment. See the SecureData Deployment Architecture diagram below summarizing these options:

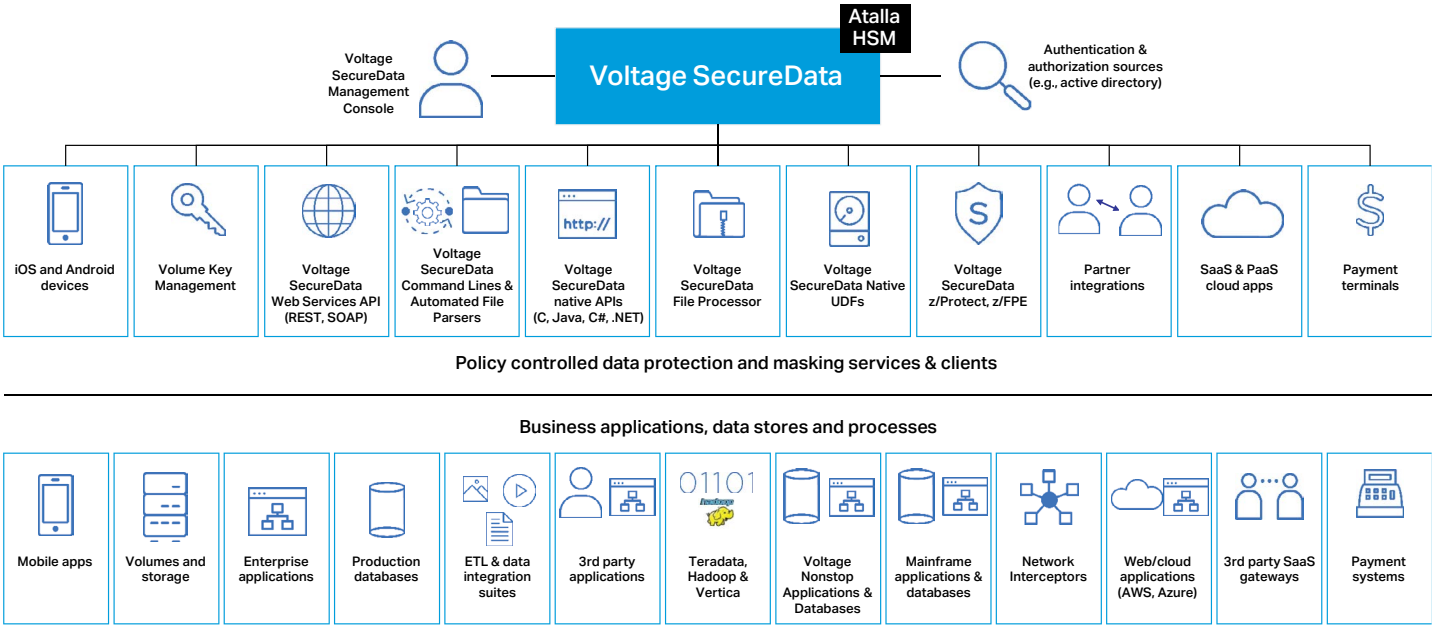

Figure 1. Voltage SecureData Architecture addresses use cases for enterprises across diverse environments.

SecureData Data Protection Options Using Hadoop

Using a data-centric security approach, it is necessary to identify the sensitive data fields within all data sources that will be transferred into Hadoop, and make a determination as to the most appropriate point to de-identify the data. The general principal is to protect the data as close to the original source as possible. Put another way, the best place to protect data in Hadoop is before it gets to Hadoop. This is not always possible, and encryption/tokenization can also be evoked during an ETL transfer to a landing zone, or from the Hadoop process transferring the data into HDFS.

Once the secure data is in Hadoop, it can be used in its de-identified state for additional processing and analysis without further interaction with the SecureData system. Or the analytic programs running in Hadoop can access the clear text by utilizing the SecureData high-speed decryption/de-tokenization interfaces with the appropriate level of authentication and authorization.

If processed data needs to be exported to downstream analytics in the clear—such as into a data warehouse for traditional BI analysis—you again have multiple options for re-identifying the data, either as it exits Hadoop using Hadoop tools or as it enters the downstream systems on those platforms. SecureData provides seven specific options for protecting sensitive data used in Hadoop:

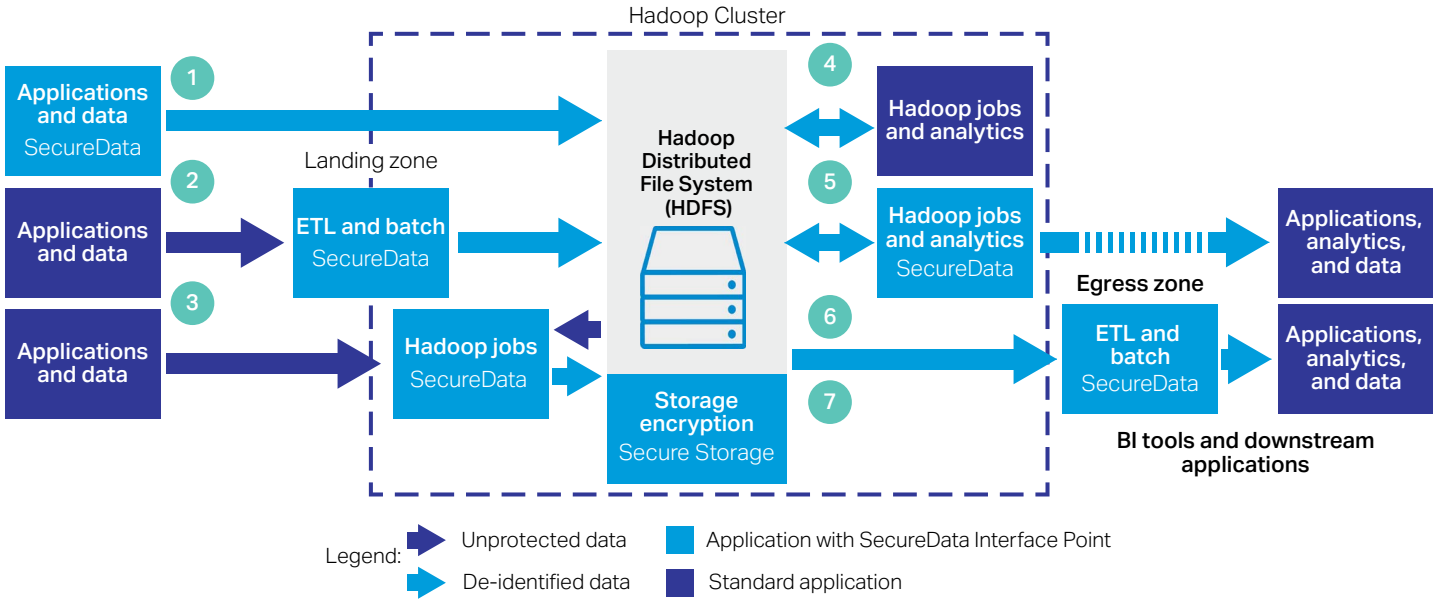

Figure 2. Hadoop Cluster

Options

1. Apply data protection at source applications
2. Apply data protection during import into Hadoop (ETL process, Sqoop)
3. Apply data protection within Hadoop (e.g., NiFi, Sqoop, Hive, MapReduce, Storm/Kafka, etc.)
4. Using de-identified data within Hadoop (e.g., Hive)
5. Using and exporting re-identified data from Hadoop (e.g., Hive)
6. Exporting data and re-identifying outside Hadoop (ETL process)
7. Using storage-level encryption within Hadoop

Option 1: Apply Data Protection at Source Applications

This is the ideal scenario, so that the sensitive data is fully protected wherever it flows, including Hadoop. It also helps ensure that the Hadoop system is not brought into scope for PCI and other compliance policies.

It requires access to the source applications outside the Hadoop environment to add the SecureData interface calls for encryption and tokenization. But once protected, this information can be transferred freely into HDFS tools and then be used "as is" for many analytic tasks (see option 4), or selectively re-identified when in-the-clear data is required (see options 5 and 6).

To facilitate the discussion, we have constructed two example data base tables (each one record in length), the first showing multiple fields related to a credit card transaction, the second showing credit score indexed by social security number (SSN). The records are coded in red to designate unprotected, in-the-clear data.

ID	Name	Street	City	State	Post code	Phone	Email	Birthday	Credit Card	SSN
1	Tyshawn Medhurst	Verl Plaza	New Lianemouth	LA	44638	(405)920-0731	**tyshmedhurst@gmail.com**	3/2/1977	5225629041834450	675-03-4941

Table 1.

SSN	Score
675-03-4941	621

Table 2.

Email, birthdate, credit card number and social security number have been determined to be sensitive information that must be protected (marked in yellow). Using FPE technology, email, birthdate and SSN are encrypted as close to the source of the data as possible, using whatever SecureData platform interface is required. SST is used to tokenize the credit card number. This results in the following de-identified data records:

ID	Name	Street	City	State	Post code	Phone	Email	Birthday	Credit Card	SSN
1	Tyshawn Medhurst	Verl Plaza	New Lianemouth	LA	44638	(405)920-0731	**F91VRPV1xfas@hpF63.uk2**	8/2/1905	5225629286144450	246-07-4941

Protected Table 1.

SSN	Score
246-07-4941	621

Protected Table 2.

Data from these protected tables are available to be loaded into Hadoop as needed; including subsets of the fields and/or joined data using the social security number as the index, even though that is an encrypted field, for example:

To facilitate the discussion, we have constructed two example data base tables (each one record in length), the first showing multiple fields related to a credit card transaction, the second showing credit score indexed by social security number (SSN). The records are coded in red to designate unprotected, in-the-clear data.

ID	Name	State	Zip code	Email	Birthday	Credit Card	SSN	Score
1	Tyshawn Medhurst	LA	44638	**F91VRPV1xfas@hpF63.uk2**	8/2/1905	5225629041834450	246-07-4941	621

FPE and SST preserve referential integrity, which means protected data can be consistently referenced and joined across tables and data sets.

Option 2: Apply Data Protection during Import into Hadoop (e.g., ETL Process or Sqoop Ingestion)

This is a good option that does not require interfacing at the source applications—the data is protected as it enters Hadoop. This can either be achieved by interfacing with SecureData from traditional ETL and batch tools in a landing zone outside Hadoop, or by a Hadoop-resident import program such as Sqoop.

As part of the SecureData Hadoop Installation Kit, Developer Templates and documentation are included to guide the programmer in properly calling the SecureData interfaces from common Hadoop programs including Sqoop, MapReduce and Hive.

In this case, Sqoop would be one of the preferred methods of ingesting data into the Hadoop instance. Sqoop can connect to an SQL database and load data directly into HDFS. While Sqoop is not a full ETL tool, it does provide efficient and fast Extract and Load functionality. Although Sqoop does not provide transformation functionality as a standard capability, Security–Data Security has developed a unique approach that enables encryption operations to be performed during a Sqoop import job.
To secure data during the ingestion, the sensitive fields to be protected are identified, and the format to be used for de-identification is defined. A simple excerpt from the SecureData configuration file for protecting credit card data would be as follows:

Sqoop.field.0.column.name = credit_card
Sqoop.field.0.format.name = cc-sst-6-4

That would convert the unencrypted credit card number stored in the data store to follow the encryption settings for policy 'cc-sst-6-4', which encrypts the middle digits of the credit card number while leaving the first six and last four in their original state. After generation of the import classes, the system then automatically de-identifies the information in the "credit_card" column when loading the information into Hadoop.

With this integration, the source information stored in the database that Sqoop is accessing might be this:

ID	Name	State	Zip code	Email	Birthday	Credit Card	SSN	Score
1	Tyshawn Medhurst	LA	44638	**tyshmedhurst@gmail.com**	3/2/1977	5225629041834450	675-03-4941	621

While the actual data being imported into Hadoop, assuming de-identification of the four sensitive fields, would be this:

ID	Name	State	Zip code	Email	Birthday	Credit Card	SSN	Score
1	Tyshawn Medhurst	LA	44638	**F91VRPV1xfas@hpF63.uk2**	8/2/1905	5225629041834450	246-07-4941	621

When using Sqoop, it is important you segregate the Hadoop processing in its own landing zone. Clear data will temporarily exist in memory, but only protected data will be written into HDFS.

Option 3: Apply Data Protection within Hadoop (e.g., NiFi, Sqoop, Hive, MapReduce, Storm/Kafka, etc.)

This option utilizes the SecureData interfaces running directly within Hadoop jobs. This provides the opportunity to integrate with other Hadoop pre-processing tasks, as well as protecting data fields once they are identified in Hadoop (such as when schema are dynamically defined).

Since the MapReduce Framework is built on Java, it is easy to integrate encryption and decryption into the Map and Reduce jobs that are executed in the Hadoop cluster. SecureData Hadoop API and web services interfaces are again used, with Developer Templates supplied to assist the programmer in properly calling these SecureData interfaces.

In order to perform the encryption function, the writer of the MapReduce job needs to create a Crypto object and pass it information about the object to be encrypted and its format, so that the correct encryption for the field can be performed (i.e. Credit Card, SSN, etc). The return value is then the actual encrypted value.

String[ ] output = crypto.protectFormattedDataList(input);

The process to decrypt information is very similar—you provide the encrypted value and the decrypted information is then returned. Using our example data, we could assume that the data already stored in HDFS looks like this:

ID	Name	State	Zip code	Email	Birthday	Credit Card	SSN	Score
1	Tyshawn Medhurst	LA	44638	**tyshmedhurst@gmail.com**	3/2/1977	5225629041834450	675-03-4941	621

Once the MapReduce job is completed, there would be another file inside of HDFS that contains the encrypted information as shown below:

ID	Name	State	Zip code	Email	Birthday	Credit Card	SSN	Score
1	Tyshawn Medhurst	LA	44638	**F91VRPV1xfas@hpF63.uk2**	8/2/1905	5225629041834450	246-07-4941	621

The customer needs to be aware that with this approach, unprotected data has already entered the Hadoop system, and the original data cannot easily be removed (due to the automatic data replication and lack of a secure wipe capability within HDFS). For this reason, some customers might use a dedicated, highly controlled Hadoop cluster specifically for staging, and/or will completely rebuild the cluster once the sensitive data has been protected and transferred out.

Option 4: Using De-Identified Data within Hadoop

Once the data has been imported into Hadoop, the ideal scenario is performing all analytics and processing on the protected data, which avoids the need for decryption or de-tokenization. This is often possible by utilizing SecureData data transformation options such as leaving leading/trailing digits in the clear, maintaining data ranges, preserving date relationships, etc.

Using Hive on our sample data, you might execute the following Query:

```
SELECT sid, s.name, s.email, ,s.birth_date, s.cc, s.ssn,cs.creditscore
FROM voltage_samples JOIN voltage_sample_creditscorecs
ON (s.ssn = cs.ssn) WHERE s.id <= 10;
```

The returned data would be:

ID	Name	State	Zip code	Email	Birthday	Credit Card	SSN	Score
1	Tyshawn Medhurst	LA	44638	**F91VRPV1xfas@hpF63.uk2**	8/2/1905	5225629041834450	246-07-4941	621

The consistency of the data is preserved, and analytical operations in Hadoop would work the same way as in the unencrypted state, including table joins. Because FPE provides high-strength encryption while it preserves business value of the data, and referential integrity across distributed data sets, the vast majority of analytics can be securely performed on the de-identified data.

Option 5: Using and Exporting Re-Identified Data from Hadoop

There will be situations in which some data fields need to be processed in their clear, unaltered form, and this is possible using the same interfaces as discussed in Options #2 and #3 above. SecureData architecture enables the distributed, high-speed re-identification of the data so as not to slow down the Hadoop processing. If required, this data can be streamed to downstream applications for additional processing.

Taking the same HIVE example used in #4 above, only this time we assume that clear text is required, we use access functions to re-identify the information, using the Developer Template for HIVE UDF, as follows:

```
SELECT s.id, s.name, accessdata(s.email, 'alpha'), accessdata(s.
birth_date, 'date'), accessdata(s.cc, 'cc'), accessdata(s.ssn, 'ssn'),
cs.creditscore
FROM voltage_sample s JOIN voltage_sample_creditscorecs
ON (s.ssn = cs.ssn) WHERE s.id <= 10;
```

This results in returning the data in its re-identified form:

ID	Name	State	Zip code	Email	Birthday	Credit Card	SSN	Score
1	Tyshawn Medhurst	LA	44638	**tyshmedhurst@gmail.com**	3/2/1977	5225629041834450	675-03-4941	621

This information can be used temporarily by the Hadoop analytic program to facilitate its computations, or passed outside Hadoop to other post-processing applications or analytic programs. This functionality is even available remotely through ODBC and JDBC, allowing customers to utilize their favorite analytic application without the need to add SecureData code on the client side.

Option 6: Exporting Data and Re-Identifying Outside Hadoop (ETL Process, Data Export or BI Import)

Hadoop data can also be bulk transferred and re-identified during ETL processes and data exports for downstream processing outside of Hadoop. As with options #1 and #2, the interface to SecureData can be evoked from virtually any standard platform.

A common use case is to perform large scale analytics in the Hadoop cluster and then transfer the reduced data set into Teradata or other computing platforms with existing fine tuned business analytic rules, where it can be processed by existing BI tools using the data either in its protected state or re-identified within the host BI environment, as needed.

Option 7: Using Storage-Level Encryption within Hadoop

Voltage SecureStorage enables production use of Transparent Data Encryption (TDE) within Hadoop to create a safe landing zone at the scale of the full Hadoop cluster, to deposit data with granular access controls. From there, data-centric security can be applied as the data is migrated into the broader cluster using tools such as MapReduce. TDE can also be used to provide the data-at-rest with granular access control for unstructured data in Hadoop.

SecureStorage also offers the option of encryption at the volume level, similar to other "data-at-rest" Hadoop encryption solutions available in the industry today. It is not necessary for any fields already protected by the data-centric security options listed above, of course, but is very useful as an extra level of defense for the entire data set stored in Hadoop, particularly unstructured data or sensitive data you might not yet be aware of. As discussed above, this reduces data exposure when cluster drives are disposed, repaired or re-purposed, but does not protect the data if accessed while the disk is running live within the system.

The big advantage of SecureStorage volume-level encryption versus these other Hadoop tools is that it uses the same key servers and encryption infrastructure as SecureData data field-level products, enabling simplified and consistent key management across your organization. With Stateless Key Management, there is no need for key backup, and no possibility of key loss.

Conclusion

Hadoop is an exciting new set of technologies whose unique architecture provides the ability to efficiently process huge sets of data. However, by its nature it is less secure than traditional enterprise computing environments, while presenting a particularly high-value target for hackers and thieves. The only way to ensure sensitive information is protected in Hadoop is to protect the data itself, ideally before it enters the Hadoop data store. This protection needs to prevent the data from unauthorized access while at-rest, in-use and in-motion.

SecureData data-centric security platform provides unique benefits that can provide this protection while still maintaining its value. Specifically, SecureData provides:

- The ability to protect data as close to its source as possible.
- Support for encryption, tokenization and data masking protection techniques.
- Data is usable for most applications in its de-identified state.
- **The ability to securely re-identify data when required—** only by authorized users and applications.
- Protection techniques backed by security proofs and standards.
- High performance, high scalability well-matched with Hadoop speeds.
- **Broad platform and application support—**inside and outside Hadoop.

Learn more at
https://software.microfocus.com/products

USE CASE

Global Telecommunication Company

A global communications company wanted to use Hadoop to analyze massive customer data sets, social media and communications feeds for patterns of behavior in order to detect fraud and enhance customer service. Voltage FPE technology was used to de-identify sensitive data in several hundred million customer records from Teradata and IBM Mainframes as it was ingested into Hadoop in under 90 seconds.

USE CASE

Health Care Insurance Company

A leading health care insurance company wanted to open their massive, previously untapped data sets to their Hadoop developers to enable research, discovery and innovation through developer hackathons. They also had the objective to automate multiple high value use cases, such as identification of prescription fraud, previously hampered by manual processes. Voltage FPE technology enabled field-level de-identification of sensitive ePHI and PII data across a 1000-node distribution.

USE CASE

Global Financial Services Company

A global financial services firm needed to adopt Hadoop to improve its fraud detection capabilities, analyzing customer transaction patterns across hundreds of millions of consumers. Data de-identification in-stream during ETL ingestion with Informatica enabled secure analytics across fields containing dates, date ranges, card-holder data, and consumer personal data without exposing the live data. After Hadoop processing, the de-identified data is transferred into their traditional BI tools for additional analysis. The data remains protected throughout with end-to-end field level encryption and tokenization, avoiding compliance and risk challenges, while enabling new approaches to reducing fraud.

NIST Special Publication 800-38G

Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption

Morris Dworkin

COMPUTER SECURITY

Voltage FPE facilitates the targeting of encryption to sensitive information, as well as the retrofitting of encryption technology to legacy applications, where a conventional encryption mode might not be feasible because it would disrupt data fields/ pathways. FPE has emerged as a useful cryptographic tool, whose applications include financial-information security, data sanitization, and transparent encryption of fields in legacy databases.Source: US Government NIST SP 800- 38G Standard

Contact us at:
www.microfocus.com

Like what you read? Share it.