Enterprise adoption of big data is maturing as both a strategy and an architectural destination in the journey toward a modern data architecture. The majority (70 percent) of companies already involved in this transformation are using Hadoop for data discovery, data science, and big data projects.1 There is a big impediment to big data moving into production, however, especially for those built around Hadoop—an open source framework for storing and running applications on commodity hardware—due to security concerns. As top use cases include data science/big data projects, real-time analytics for operational insights, and centralized data acquisition or staging for other systems, massive quantities of data including highly sensitive payment card data (PCI), personally identifiable information (PII), and protected health information (PHI) are being moved into these environments. The fear is not unreasonable; the risk is high given what cyber attackers are after and the extreme damages that may result from a successful data breach.

The challenge
Consider the steps taken before a data breach. The first step attackers take would be to construct a map laying out the network of the target organization, identifying what and where systems are located. This is an important step as the goal is typically not to disrupt the company, but rather to set up mechanisms to acquire data over as long a run as possible—and monetize it. Preferably they would want to include hooks so that they get periodic data updates from their target.

Now when an IT organization builds a big data environment, the target has already done a lot of work for the attacker. With big data environments the enterprise will have actually created a single location for all the valuable data assets attackers are seeking. Many times the data is even cleaned and consolidated so that attackers can conveniently monetize the data even faster.

Making matters even worse is the fact that when Hadoop was developed, security was never a concern for the early developers. Their objective was to develop a platform that can scale to huge volumes of data and process this data in extremely fast ways. So when Hadoop finally made its way from a research project into the business world, security components were bolted on to make the system more manageable from a security perspective.

However, all those security add-ons to Hadoop are only securing the perimeter and not the sensitive data inside Hadoop. It is a well-known fact that while perimeter security is

---

an important aspect in any company’s security portfolio, it is also increasingly insufficient. “On average it takes an organization 200 days to find out it has been compromised.” — Dick Bussiere, Tenable Network Security. This leaves the most sensitive data assets at risk for over 200 days during which time the cyber attackers can funnel the most valuable data out of their target, with the scale of the breach growing every day.

With this in mind companies are trying to avoid being the lead article in the news by going slow in adoption of promising new strategies and technologies such as big data adoption. A major data breach can severely damage a business for years due to fines, loss of customer confidence and trust, loss of stakeholders no longer willing to invest in the company due to the bad publicity, and damage to the brand and corporate reputation. According to a 2015 Ponemon Institute study on the cost of cybercrime — recovery from such an incident on average can cost $3.79M — can take years to emerge from, or may not be possible; a data breach could put a company completely out of business.¹

The way forward

With attacks becoming more frequent and larger in size, it is natural that companies are putting more and more effort into protecting their most valuable assets—the data itself—with data-centric security technology. Data-centric security renders the data unusable for attackers in the event of a breach.

Apache Hadoop

Hadoop is an open-source software framework for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power, and the ability to handle virtually limitless concurrent tasks or jobs.


The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly available service on top of a cluster of computers, each of which may be prone to failures.

Apache Hadoop — [hadoop.apache.org](http://hadoop.apache.org)

A simple and efficient way to do this is to apply Format-Preserving Encryption.

Format-preserving encryption (FPE) has been around for a while and is in the process of being recognized by important standards bodies such as NIST (Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption – SP800-38G⁴). This standard defines two modes of format-preserving encryption, identified as FF1 and FF2 in the publication. It is important when applying FPE that the vendor providing the solution has been vetted by third parties such as the NIST standardization process. Not all encryption algorithms are perfectly safe, but the peer review that is part of the NIST standardization process can detect such problems and thus ensure the data will be adequately protected.

---


Advantages of FPE

But what actually is FPE? It is a form of AES encryption that has been in use for some time, mainly in encrypting disc drives and communications between end points such as SSL/TLS and VPNs. But unlike AES, which encrypts data into a large block of random numbers and letters, FPE encrypts the original value into something that looks like the original. For example, a credit card number still looks like a credit card number.

Another advantage of FPE is that certain data elements can be preserved during the encryption process. An example of this is to preserve the bank routing number (the first six digits) in a credit card number so that the inherent value of this information can be maintained for analytical purposes.

As with any encryption the same plaintext will encrypt to the same ciphertext. This allows referential integrity to be maintained. This means that when combining multiple data sources into the big data environment where all data is encrypted, all the links between the data will work just as well as before. A social security number is frequently used to link data records from different sources. So after encrypting those values with FPE, the linkage functions exactly as it did on the original unencrypted data.

How to implement FPE in a big data environment

The most frequent use is to encrypt data during the ingestion process when data is passed from the source system to the big data ecosystem. This is especially critical for Hadoop environments because once data is ingested, it is very difficult to identify on which disc drive the data is residing. One of the reasons is that Hadoop automatically keeps three copies of the data by default. A traditional approach to security would be to use disc-drive or volume-level encryption with Hadoop. However, the experience many people have reported with this approach is a dramatic performance penalty. While this level of performance penalty can be workable with a very small Hadoop cluster, it is cost-prohibitive for larger Hadoop instances.

But integrating FPE into the ingestion process can be a simple addition to existing process work flows built around Sqoop5 for Hadoop or inside Informatica, Data Stage, and other tools that extract and transform data from a source system, such as a regular database, into the big data environment. This approach puts the encryption as close to the data source as possible. When data is actually streamed into Hadoop, integration points such as Flume6 and Storm7 allow data protection to happen in-stream, before entering the actual Hadoop ecosystem.

However, this approach is not always feasible or suited for every situation. Sometimes people prefer a landing-zone approach where they can deposit data directly into Hadoop. In this scenario, there are two typical ways to address the need for data-centric security.

The first approach is to utilize Transparent Data Encryption (TDE) in Hadoop to provide file/folder-level encryption. While analysts frequently discourage the use of TDE in production environments due to the lack of proper key management or security of the key manager, there are solutions available in the market that provide an alternative key manager for TDE that actually delivers the proper security and audit elements needed for proper key management. From a workflow viewpoint, users can load data directly into Hadoop, where the data is protected at rest by encryption and access controls for the encryption zone in TDE. Once the data is

---

5 Apache Sqoop – http://sqoop.apache.org/
6 Apache Flume – https://flume.apache.org/
7 Apache Storm – http://storm.apache.org/
Protecting Your Data against Cyber Attacks in Big Data Environments | Reiner Kappenberger

ingested, a MapReduce\(^8\) job can then be utilized to protect the data on a field level with FPE and transfer the output file to the larger cluster so that the file with FPE-protected data is accessible by other users.

The second approach being frequently utilized is where the data is actually deposited outside of the big data environment (e.g., to a Linux server). Again, data-at-rest encryption is utilized to protect the sensitive data coming in. Using elements such as a file processor to take the unprotected input file and process each data record and apply FPE creates an output file with the protected data. This output file can then be moved into the big data system (Hadoop, Vertica, Teradata, etc.) in its FPE-protected form as before.

Figure 1 is an example of what protected data can look like when applying format preserving encryption and tokenization to the data. It shows both the original values (red table) and the protected data (green table).

**Securing the Internet of Things**

When big data is used within an Internet of Things (IoT) environment, bulk loading the data is not feasible as data is continuously streamed from devices into big data for analysis and decision-making. Typically in those environments, solutions like Flume and Storm/Kafka\(^9\) are used with Hadoop. By integrating format-preserving encryption and tokenization into those streams, data can be protected during the ingestion and decision-making process.

As previously outlined, referential integrity can be preserved with FPE. So, once all data is protected, the analytics can almost always be done with the protected data. Frequently analysts state that they need the data in the clear to perform their analytics. However, when asked why data needs to be in the clear, the answers are usually vague. The main reason mentioned is the need for a cleartext reference for search. It’s a common misconception that the information in big data would need to be decrypted to perform a search. In fact, executing a search against the FPE-protected data would actually yield the same result without exposing any sensitive data during query execution.

Since elements or sub-fields from the original data format can be exposed (e.g., the first six digits of a credit card number), analytics can still be performed on encrypted data where certain elements of the original data are needed. In the above example a retailer could still identify the revenue generated during a promotion that was done with a specific credit card issuer.

**Implementing data-centric security in the data lake**

Data lakes, however, are not a single environment. A data lake is a bigger ecosystem\(^10\) that is frequently comprised of multiple data warehouse solutions (e.g., Hadoop, Vertica, or Teradata), extraction and transformation tools (e.g., Informatica, Data Stage, SAS, etc.), and analytical front ends. So, one of the most important aspects of implementing a data-centric security strategy is to identify integration points.

The best approach for integration is to select an encryption vendor that has APIs available that either are already integrated into the desired tools or can be integrated on the platforms where protection is needed. Ultimately the goal would be to protect data as early in the processing stage as feasible and only retrieve the actual result as close to the real need as possible.

There are multiple reasons for this. First, an analyst typically does not need to be able to identify the actual individual behind the data analysis. An analyst should identify trends, groups of individuals, patterns, and the like. However, the analyst does not actually act on the result set. An example from the medical field would be the need to find certain patients that have responded in a particular manner to a medication where an alternative treatment might be a better solution. The analyst would run the investigation to find the group of individuals but will not contact an individual in the study. That action would be performed by another person in the organization whose role may be to contact the doctor and make a suggestion for certain patients about a different treatment option.

The second reason is that data lakes frequently consist of a traditional data warehouse environment (e.g., Vertica, Teradata, Greenplum, Hana) in connection with Hadoop and its tools. This means that protected data must be able to move freely between the environments without the need to repeatedly decrypt and re-encrypt the data between each environment. Data should be moved with standard tools without the need to know whether the data is actually protected.

---

The key to effective encryption

Encryption is not just about the actual encryption techniques used; it is as much about key management and authentication technologies. In order to encrypt information, an encryption key is needed. There is a saying that “key management is hard.” So, when selecting an encryption solution, it’s very important to consider the functionality and approach to key management.

Traditional key management solutions are designed to store keys in a secure environment. However, this does create challenges; one of the biggest is that keys that are stored must be replicated between the different key servers. This approach creates several issues. The first is that, at some point, the key servers will spend more time synchronizing the key stores than actually serving requests, resulting in a decrease in performance and limited scalability. Also, proper backup and restore procedures need to be implemented because when a key is lost, all data protected with this key is rendered unusable.

A way to avoid this key replication problem is to use a stateless key management solution. With stateless key management, keys are generated from a master key (or root key) that often resides in a hardware security module (HSM) for maximum security. With such a solution there is no physical key store (typically implemented with a database in the key manager) that needs to be moved between the key servers and properly maintained. Every time a key is requested, the key can be regenerated from this root key. This typically is referred to as Identity-Based Encryption (IBE), a form of ID-based encryption that means any party can generate a key from a known identity value such as an ASCII string (e.g., a user’s email, address, name, etc.).

Using such key management functionality simplifies backup and restore processes: there are no keys actually stored in the key manager. Even more importantly, this allows the system to scale in a more linear fashion as there is no communication required between the different key servers. When additional capacity is needed—such as when expanding Hadoop—bringing up additional key managers is a simple procedure like adding more nodes in Hadoop.

Where this need for stateless key management becomes even more critical is when the solution is operated from multiple locations. This can either be a primary/backup data center configuration or even a Hadoop instance that is distributed across multiple locations for better availability. Key replication is never instantaneous, but when using IBE to derive keys on the fly, the keys are always available and will never be lost.

Conclusion

The bottom line is that in order to properly protect a big data environment against attacks, one needs to look further than just the traditional perimeter security provided by either the big data vendor or other traditional infrastructure tools. Protecting the data itself, using data-centric security, provides a way to protect data against attacks while still maintaining the value of the data and allowing analytics without imposing a security-performance penalty. With standards-recognized format-preserving-encryption techniques, sensitive data is protected not only at rest in big data environments but also in motion and in use for analytics. And in the event of a data breach the cyber attackers gain nothing of value. Stateless key management in combination with field-level, format-preserving encryption, enables the secure portability of data throughout Hadoop and big data ecosystems.

Additional References


About the Author

Reiner Kappenberger has over 20 years of computer software industry experience focusing on encryption and security for big data environments. His background ranges from device management in the telecommunications sector to GIS and database systems. He holds a Diploma from the FH Regensburg, Germany, in computer science. He may be reached at reiner.kappenberger@hpe.com.