Securosis

Securing Big Data: Security Recommendations for Hadoop and NoSQL Environments

Version 1.0

Released: October 12, 2012

Author’s Note

The content in this report was developed independently of any sponsors. It is based on material originally posted on the Securosis blog but has been enhanced, reviewed, and professionally edited.

Special thanks to Chris Pepper for editing and content support..

Licensed by Vormetric

Vormetric is the leader in enterprise encryption and key management. The Vormetric Data Security solution provides a single, manageable and scalable solution to encrypt any file, any database, any application, anywhere it resides—without sacrificing application performance or creating key management complexity.

Some of the largest and most security conscious organizations and government agencies in the world, including 16 of the Fortune 25, have standardized on Vormetric Data Security technology to provide strong, easily manageable data security. Vormetric Data Security technology has been selected by IBM as the only database encryption solution for DB2 and Informix on Linux™, Unix® and Windows; by Symantec to provide the Symantec Veritas NetBackup™ Media Server Encryption Option.

For more information, visit: vormetric.com

Copyright

Table of Contents

Executive Summary

This paper examines security for "big data" environments, reviewing built-in protections and weaknesses of these systems. Our goal is to educate big data users on security problems they face with pragmatic advice on how to secure these environments. During this research two facts became abundantly clear. First, big data projects are common — almost the norm — within the enterprises we spoke with. They have embraced the technology and pushed vast amounts of data into these clusters. Second, most have implemented little or no security beyond user passwords.

Our examination of different big data implementations shows that security features are sparse and aftermarket offerings are not fully tailored to these clusters. It appears that in their rush to implement highly scalable low-cost clusters for data analysis, security is largely an afterthought. Our findings show these deployments to be largely insecure, and wholly reliant on network and perimeter security support.

The good news is that several critical security concerns can be addressed by a handful of security measures. Further, as many clusters are being deployed within virtual and cloud environments, they can leverage vendor supplied management tools to address many of the operational security issues. While these measures cannot provide fail-proof security, a reasonable amount of effort can make it considerably more difficult to subvert systems or steal information.

Summary of Recommendations

Big data clusters share most of the same vulnerabilities as web applications and traditional data warehouses. Concerns over how nodes and client applications are vetted before joining the cluster, how data at rest is protected from unwanted inspection, privacy of network communications, and how nodes are managed, were our principal concerns. The security of the web applications that front big data clusters is equally important, but these challenges also exist outside big data clusters so they are outside the scope of this paper. Our base recommendations are as follows:

- Use Kerberos — included in Hadoop — to validate nodes and client applications before admission to the cluster, and to validate application requests for MapReduce (MR) and similar functions.
- Use file/OS layer encryption — to protect data at rest, ensure administrators or other applications cannot gain direct access to files, and prevent leaked information from exposure.
- Use key/certificate management — you can't store keys and certificates on disk and expect them to be safe. Use a central key management server to protect encryption keys and manage different keys for different files.
- Validate nodes during deployment — through virtualization management, cloud provider facilities, or third-party products such as Chef and Puppet.
- Log transactions, anomalies, and administrative activity — through logging tools that leverage the big data cluster itself — to validate usage and provide forensic system logs.

- Use SSL or TLS network security — to authenticate and ensure privacy of communications between nodes, name servers, and applications.

This list is far from comprehensive, but these techniques protect against basic attacks with minimum disruption to applications and operations. Open issues — where solutions are either inadequate or non-existent — include the security of web applications that use big data clusters, authentication and authorization, and real-time monitoring. Details of our findings follow, along with additional recommendations and further information on outstanding issues.

Introduction

How can I secure "big data"? A simple and common question. But unfortunately there is no simple answer.

Thousands of firms are working on big data projects, from small startups to large enterprises. New technologies enable any company to collect, manage, and analyze incredibly large data sets. As these systems become more common, the repositories are increasingly likely to be stuffed with sensitive data. Only *after* companies find themselves reliant on "big data" do they ask how to secure it.

This question comes up so much, attended by so much interest and confusion, that it's time for an open discussion of big data security. To help get a better handle on the challenges we need to examine several aspects. Specifically, we will cover three things:

- **Why It's Different Architecturally:** What's different about these systems, both in how they process information and how they are deployed? We will list some specific differences and discuss how they impact data and database security.
- **Why It's Different Operationally:** We will detail operational security issues with big data platforms. We will offer perspective on the challenges of securing big data and the deficiencies of the systems available to manage it — particularly their lack of native security features.
- **Recommendations and Open Issues:** We will outline strategies for securing these data repositories, with tactical recommendations for securing certain facets of these environments. We will also highlight gaps where no good solutions exist.

Before we can offer advice on securing anything we need to agree on what we're talking about, and the definition of "big data" is still not settled. The term is so overused that it has become almost meaningless. When we talk to customers, developers, vendors, and the media, they all have their own ideas of what "big data" means. It's a complex subject, and even Wikipedia fails to capture the essence. Like art, everyone knows it when they see it, but nobody agrees on a definition.

Defining Big Data

We know big data systems can store very large amounts of data; can manage that data across many systems; and provide facilities for data queries, data consistency, and systems management.

So does "big data" mean **any** giant data repository? No. We are not talking about giant mainframe environments. We're not talking about grid clusters, proprietary MPP databases, SAN arrays, cloud-in-a-box, or even traditional data warehouses. We have had the capability to create *very* large data repositories and databases for decades. The challenge is not to manage a boatload of data — many platforms can do that. And it's not just to analyze very large data sets. Various data management platforms offer the capability to analyze large amounts of data, but their *cost* and *complexity* make them non-viable for most applications. The big data revolution is not just about new capacity frontiers for storage and analysis.

Is big data a specific technology? Can we say that big data is any HDFS/Lustre/Google GFS/shard storage system? It's not that simple — a distributed file system is a key ingredient, but big data is more than that. Is big data any MapReduce cluster? No, because even PL/SQL subsystems in traditional Oracle databases can be set up to work like MapReduce. MapReduce is a common ingredient, but other query engines are used by different flavors of NoSQL built upon Hadoop for selecting different types of data. So is big data a type of data analytics application? Actually, it's all these things and more.

When we talk to developers, the people actually building big data systems and applications, we get a better idea of what the term means. The design simplicity of these these platforms is what attracts developers. They are readily available and their relatively low deployment cost makes them accessible to a wide range of users. With all these traits combined, large-scale data analysis becomes cost effective. Big data is not a specific technology — it is a collection of attributes and capabilities.

Sound familiar? It's more than a bit like nailing down cloud computing, so we'll steal from the NIST cloud computing definition and start with essential characteristics.

Essential Characteristics

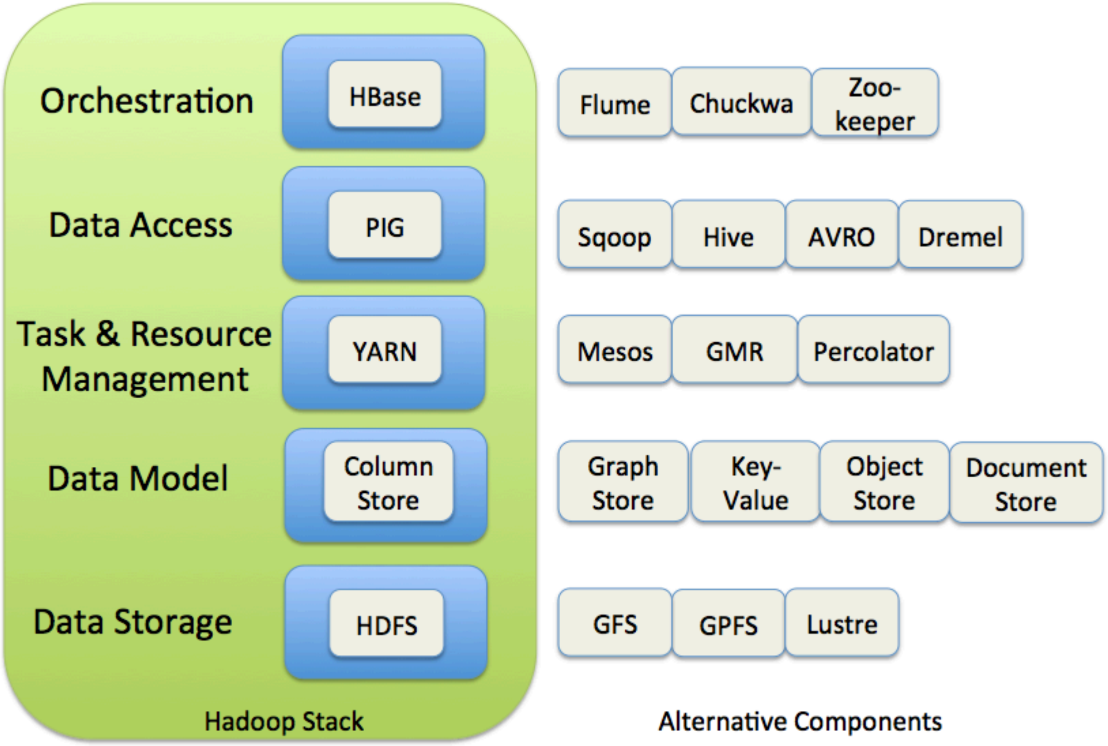
We define big data as any data repository with the following characteristics:

- Handles large amounts (a petabyte or more) of data
- Distributed redundant data storage
- Parallel task processing
- Provides data processing (MapReduce or equivalent) capabilities
- Extremely fast data insertion
- Central management and orchestration
- Inexpensive (relatively)
- Hardware agnostic
- Accessible — both relatively easy to use, and available as a commercial or open source product
- Extensible — basic capabilities can be augmented and altered

In a nutshell: big, cheap, and easy data management. The "big data" revolution is built on these three pillars — the ability to scale data stores at greatly reduced cost makes it all possible. It's data analytics available to the masses. It may or may not have traditional 'database' capabilities (indexing, transactional consistency, or relational mapping). It may or may not be fault tolerant. It may or may not include failover capabilities (redundant control nodes). It may or may not store complex data types. It may or may not provide real-time results to queries. But big data offers all the characteristics above, and it turns out that they — even without traditional database features — are enough to get useful work done.

So does big data mean the Hadoop framework? Kinda. The Hadoop *framework* (*e.g.,* the combination of HDFS, YARN, Common, etc.) is the poster child for big data, and it offers all the essential characteristics. Most big data systems use one or more Hadoop components, then replace or extend some of the basic functionality. Amazon's SimpleDB also satisfies our requirements, although it is architected differently than Hadoop. Google's proprietary BigTable architecture is very similar to Hadoop — which was designed as an open source alternative — but we exclude proprietary systems which are not widely available. Our definition leaves some grey areas but is useful for understanding why the trend is so popular. For the remainder of this document, unless we say otherwise, we will focus on the Hadoop framework and its common NoSQL variants (Cassandra, MongoDB, Couch, Riak, etc.) as together they represent the majority of customer usage. In fact we'll use Hadoop, NoSQL and 'big data' interchangeably, understanding that the definition encompasses more than just those options.

It is useful to think about the Hadoop framework as a 'stack', much like a LAMP stack. These pieces are normally grouped together but you can mix and match and add onto the stack as needed. For example, Sqoop and Hive are replacement data access services. Lustre, GFS, and GPFS are alternatives to HDFS. You can select a big data environment specifically to support columnar, graph, document, XML or multidimensional data — collectively called 'NoSQL', as they are not constrained by relational database constructs or a relational query parser — you can substitute different query engines depending on the type of data being stored. Or you can extend HDFS functionality with logging tools like Scribe. The entire stack can be configured and extended as needed. This modular approach offers great flexibility but makes security more difficult, as each option brings its own security options and deficiencies. We get big, cheap, and easy data management and processing — but the security capabilities lag behind.

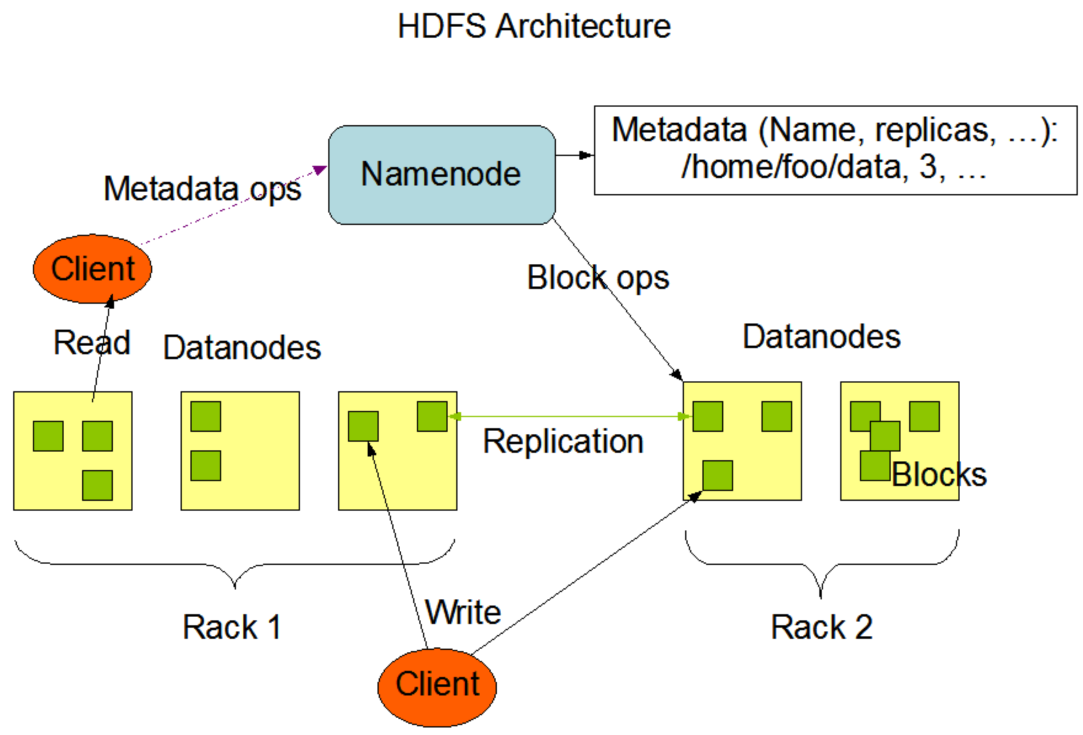


Architectural Security

We have defined big data (more or less), so now we can discuss deployment. This is critical — the architectural model and deployment options means you need a different approach to securing big data clusters. Further, many built-in security capabilities are absent from big data distributions, so the security challenges are much different than traditional databases, data warehouses, and massively parallel processing environments.

Big data is distinguished by its fundamentally different deployment model: highly distributed, redundant, and elastic data repositories enabled by the Hadoop File System. A distributed file system provides many of the essential characteristics (distributed redundant storage across resources) and enables massively parallel computation. But specific aspects of how each layer of the stack integrates — including how data nodes communicate with clients and resource management facilities — raise many concerns.

For those of you not familiar with the model, this is the Hadoop architecture:



Graph credit: Apache Software Foundation

Architectural Issues

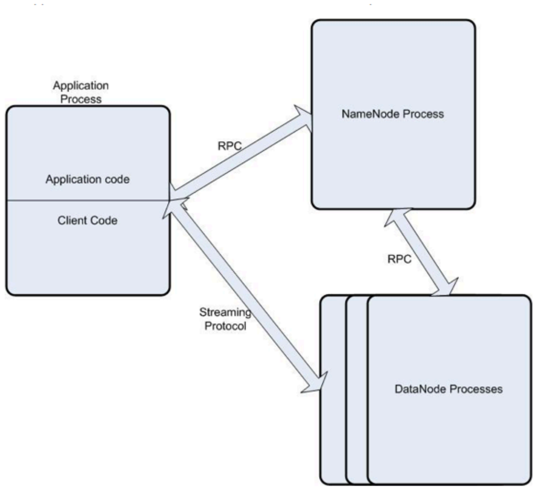
- **Distributed nodes:** “Moving computation is cheaper than moving data” is the key to big data. Data is processed anywhere resources are available, enabling massively parallel computation. This creates complicated environments with plenty of attack surface, and it is difficult to verify security consistency across a highly distributed cluster of possibly heterogeneous platforms.

- **‘Sharded’ data:** Data within big data clusters is fluid, with multiple copies moving to and from different nodes to ensure redundancy and resiliency. A ‘shard’ is a slice of data — horizontally segmented — shared across multiple servers. This automated movement to multiple locations makes it very difficult to know precisely where data is located at any moment in time, or how many copies are available. This runs counter to the traditional *centralized* data security model, where a single copy of data is wrapped in various protections until it is used for processing. Big data is replicated in many places and moves as needed. The ‘containerized’ data security model is missing — as are many other relational database facilities.

- **Data access/ownership:** Role-based access is central to most database security schemes. Relational and quasi-relational platforms include roles, groups, schemas, label security, and various other facilities for limiting user access to authorized subsets of the available data. Most big data environments offer access limitations at the schema level, but no finer granularity. It is possible to logically mimic label security and other advanced capabilities in big data environments, but that requires the application designer to build these functions into applications and data storage.

- **Inter-node communication:** Hadoop and the vast majority of distributions (Cassandra, MongoDB, Couchbase, etc.) don’t communicate securely — they use RPC over TCP/IP. TLS and SSL are rarely bundled in the big data distribution. When they are — as with HDFS proxies — they only cover client-to-proxy communication, not proxy-to-node communication.

- **Client interaction:** Clients interact with resource managers and nodes. While gateway services can be created for loading data, clients communicate directly with both resource managers and individual data nodes. Compromised clients can send malicious data or links to either service. This model facilitates efficient communication but makes it difficult to protect nodes from clients, clients from nodes, or even name servers from nodes. Worse, the distribution of self-organizing nodes is a poor fit for security tools such as gateways/firewalls/monitoring, which require a ‘choke-point’ not available in a peer-to-peer ‘mesh’ cluster.



- **NoSecurity:** Finally, and perhaps most importantly, big data stacks build in almost no security. As of this writing — aside from service-level authorization and web proxy capabilities from YARN — no facilities are available to protect data stores, applications, or core Hadoop features. All big data installations are built on the web services model, with few or no facilities for countering common web threats, (*i.e*. everything on the OWASP Top Ten list) so most big data APIs are vulnerable to well known attacks.

There are a couple architectural security issues inherent to big data. They are not specific to big data, but big data projects are inherently subject to them because of the distributed architecture, use of a simple programming model and the open framework of services. To add security capabilities into a big data environment, the capabilities need to scale with the data. Most ‘bolt-on’ security do not scale in this way, and simply cannot keep up. Because the security controls are not built into the products, ‘bolt-on’ security creates a classic impedance mismatch between Hadoop/NoSQL environments and aftermarket security tools. Most security vendors have adapted their existing offerings as well as they can — generally applying a control point where data and commands *enter* the cluster — but not *within* the cluster. Only a handful of traditional security products can fully integrate and dynamically scale with Hadoop clusters.

> Most security tools fail to scale and perform with big data environments.

Operational Security

When we talk about big data security, we are referring to both data *and* infrastructure security. We want to protect the application (*database* if you prefer) that manages data, in order to protect the information under management. If an attacker can access data directly, bypassing the database management system, they will. Barring a direct path to the information, they will look for weaknesses or ways to subvert the database application. When we talk about database security it's important to remember that we need to protect both data *and* infrastructure.

Beyond the architectural security issues endemic to Hadoop and similar platforms, they lack many of the common security controls IT management teams expect from other data management platforms. That includes "turning the dials" on configuration management and access controls, as well as 'bolt-on' capabilities such as auditing and security gateways. Whatever NoSQL variant you choose is likely to offer one or two security controls. It might be a web proxy, network data encryption, or full bi-directional authentication of administrative access. But generally it's a single security feature, not a comprehensive set of tools. Worse, security is always optional — not enabled by default. Even Hadoop web consoles allow access without any form of authentication!

The following is an overview of the most common threats to data management systems. Data stored in with big data clusters is commonly regulated by law, and needs to be protected in accordingly. Attacks on corporate IT systems and data theft are prevalent, and large distributed data management systems provide a tempting target. Big data offers all the same weak points we know from traditional IT systems, but we have technologies that address common threats so there is no need to reinvent the wheel. The trick is selecting options that work with big data. Cluster administrators should consider security controls for each of the following areas when setting up or managing a big data cluster:

- **Data at rest protection:** The standard for protecting data at rest is encryption, which guards against attempts to access data outside established application interfaces. With traditional data management systems we worry about people stealing archives or directly reading files from disk. Encrypted files are protected against access from users without encryption keys. Replication effectively replaces backups for big data, but that does not mean a rogue administrator or cloud service manager won't create their own. Encryption protects data copied from the cluster. One or two obscure NoSQL variants provides encryption for data at rest but most do not. Worse, most available encryption products lack sufficient horizontal scalability and transparency to work with big data. This is a critical issue.

- **Administrative data access:** Each node has at least one administrator with full access to its data. As with encryption we need a boundary or facility to provide separation of duties between different administrators. The requirement is the same as on relational platforms — but big data platforms lack their array of built-in facilities, documentation, and third-party tools to address this requirement. Unwanted direct access to data files or data node processes can be addressed through a combination of access controls, separation of roles, and encryption technologies — but out-of-the-box, data is only as secure as your least trustworthy administrator. It's up to the system designer to select controls to close this gap.

- **Configuration and patch management:** With clusters of servers it is common to have nodes running different configurations and patch levels as new nodes are added over time. Or if you use dissimilar OS platforms in the cluster, determining what constitutes equivalent patch revision levels can be difficult. Existing configuration management tools work for underlying platforms, and HDFS Federation will help with cluster management, but careful planning is still necessary. We will go into detail about how to accomplish this later, under Recommendations. The cluster may tolerate nodes cycling without loss of data or service interruption but reboots can still cause serious performance issues, depending on which nodes are affected and how the cluster is configured. The upshot is that people worry about user complaints and don't patch. Perhaps you have heard that one before.

- **Authentication of applications and nodes:** Hadoop can use Kerberos to authenticate users and add-on services to the Hadoop cluster. But a rogue client can be inserted onto the network if a Kerberos ticket is stolen or duplicated — perhaps using credentials extracted from virtual image files or snapshots. This is more of a concern when embedding credentials in virtual and cloud environments, where it is relatively easy to introduce an exact replica of a client app or service. A clone of a node is often all that's needed to introduce a corrupted node or service into a cluster. It is easy to impersonate a cluster node or service, but that requires an attacker to compromise your management plane or obtain a client backup. Kerberos improves security significantly but you still need to be careful. We know it is a pain to set up, but strong authentication of nodes is a principal security tool for keeping rogue servers and requests out of your cluster.

- **Audit and logging:** If you suspect someone has breached your cluster, can you detect it? How could you do this? You need a record of activity. One area which offers a variety of add-on capabilities is logging. Scribe and LogStash are open source tools that integrate into most big data environments, as do a number of commercial products. So you just need to find a compatible tool, install it, integrate it with other systems such as SIEM or log management, and then actually *review* the results. Without actually looking at the data and developing policies to detect fraud, logging is not useful.

- **Monitoring, filtering, and blocking:** There are no built-in monitoring tools to look for misuse or block malicious queries. In fact there is no consensus on what a malicious big data query looks like — aside from crappy MapReduce scripts written by bad programmers. It is assumed that you will authenticate clients through Kerberos if you care about security, and MapReduce access is gated by digest authentication. Several monitoring tools are available for big data environments, but most review data and user requests at the API layer. The problem is that these solutions require a 'choke-point' — a path through which all client connections must pass. Our own David Mortman called these "after-market speed regulators" because the security bottleneck inherently limits performance. Most deliver the basic security value they claim, but require you to alter your deployment model or to forgo scaling — or both.

- **API security:** The APIs for big data clusters need to be protected from code and command injection, buffer overflow attacks, and every other web services attack. Most of the time this responsibility falls upon the application(s) that use the cluster, but not always. Common security controls include integration with directory services, mapping OAuth tokens to API services, filtering requests, input validation, managing policies across nodes, and so on. Some of the APIs even work without authentication, so people still haven't yet acknowledged the problem. Again, there are a handful of off-the-shelf solutions to help address API security issues, but most are based on a gateway that funnels users and all requests through a single interface. There are many important API issues, but they are beyond the scope of this paper.

In summary, there are many ways to build out a big data cluster, but few relevant and usable security tools. You can address the most glaring security weaknesses with a combination of built-in authentication services and add-on security products, without wrecking performance or scalability.

Technical Recommendations

The following are out security recommendations to address the issues we discussed in the previous sections. A couple key features we recommend are core to Hadoop. Several that are provided we can't recommend as what's provided is not well designed, so you'll need to gauge for yourself if they are helpful or not. This compels the use of 3rd party security tools not provided within the Hadoop framework to close the gaps. But not all open-source and commercial 'solutions' are up to the task, so careful selection of both technology and deployment model is required. Many vendors claim to offer big data security, but really just the same products they offer for other back-office systems and relational databases. Those products might function in a big data cluster by restricting functionality, performance, or both. Security controls should be architecturally and environmentally consistent with the cluster architecture — not in conflict with the essential characteristics we mentioned earlier.

Requirements

Any security control used for big data must meet the following requirements:

1. It must not compromise the basic functionality of the cluster.

2. It should scale in the same manner as the cluster.

3. It should not compromise essential big data characteristics.

4. It should address a security threat to big data environments or data stored within the cluster.

Our Recommendations

In the end, our big data security recommendations boil down to a handful of standard tools which can be effective in setting a secure baseline for big data environments:

1. **Use Kerberos for node authentication:** Kerberos is effective for validating inter-service communicate and help keep rogue nodes and applications out of the cluster. And it can help protect web console access, making administrative functions harder to compromise. We know Kerberos is a pain to set up, and (re-)validation of new nodes and applications requires additional overhead. But without bi-directional trust establishment, it is too easy to fool Hadoop into letting malicious applications into the cluster or accepting malicious nodes — which can then add, alter, and extract data. Kerberos is one of the most effective security controls at our disposal, and it's built into the Hadoop infrastructure, so use it.

2. **Use file layer encryption:** File encryption protects against two attacker techniques for circumventing application security controls. Encryption protects data if malicious users or administrators gain access to data nodes and directly inspect files, and renders stolen files or copied disk images unreadable. While it may be tempting to rely upon encrypted SAN/NAS storage devices, they don't provide protection from credentialed user access, granular protection of files or multi-key support. And file layer encryption provides consistent protection across different platforms regardless of OS/platform/storage type, with some products even protecting encryption operations in

memory. Just as important, encryption meets our requirements for big data security — it is transparent to both Hadoop and calling applications, and scales out as the cluster grows. Open source products are available for most Linux systems; commercial products additionally offer external key management, trusted binaries, and full support. This is a cost-effective way to address several data security threats.

3. **Use key management:** File layer encryption is not effective if an attacker can access encryption keys. Many big data cluster administrators store keys on local disk drives because it's quick and easy, but it's also insecure as keys can be collected by the platform administrator or an attacker. Use key management service to distribute keys and certificates; and manage different keys for each group, application, and user. This requires additional setup and possibly commercial key management products to scale with your big data environment, but it's critical. Most of the encryption controls we recommend depend on key/certificate security.

4. **Deployment validation:** Deployment consistency is difficult to ensure in a multi-node environment. Patching, application configuration, updating the Hadoop stack, collecting trusted machine images, certificates, and platform discrepancies all contribute to what can easily become a management nightmare. The good news is that most big data clusters are deployed in cloud and virtual environments. You can leverage tools from your cloud provider, your hypervisor vendor, and third parties (such as Chef and Puppet) to automate pre-deployment tasks. Machine images, patches, and configurations should be fully updated and *validated* prior to deployment. You can even run validation tests, collect encryption keys, and request access tokens before nodes are accessible to the cluster. We also recommend use of the service-level authentication built into Hadoop to help segregate administrative responsibilities. Building the scripts and setting up these services takes time up front, but pays for itself in reduced management time and effort later, and ensures that each node comes online with baseline security in place.

5. **Log it!:** To detect attacks, diagnose failures, or investigate unusual behavior, you need a record of activity. Unlike less scalable data management platforms, big data is a natural fit for collecting and managing event data. Many web companies start with big data specifically to manage log files, and most SIEM and log management products embed big data capabilities for log management. There is no reason *not* to add logging onto your existing cluster. It gives you a place to look when something fails, or if someone thinks you might have been hacked. Without an event trace you are blind. Logging MapReduce requests and other cluster activity is easy, and the increase in storage and processing demands is small, but the data is indispensable when you need it.

6. **Use secure communication:** Implement secure communication between nodes, and between nodes and applications. This requires an SSL/TLS implementation that actually protects all network communications rather than just a subset. This imposes a small performance penalty on transfer of large data sets around the cluster, but the burden is shared across all nodes. Cloudera offers TLS, and some cloud providers offer secure communication options as well; otherwise you will likely need to integrate these services into your application stack.

When we speak with big data architects and managers, we hear their most popular security model is to hide the entire cluster within their infrastructure, and hope attackers won't notice it. But these repositories are now common *and* web accessible. Weak security makes them very attractive targets. Consider the recommendations above a minimum set of preventative security measures. These are easy to recommend — they are simple, cost-effective, and scalable, and they addresses real security deficiencies with big data clusters. Nothing suggested here harms performance, scalability, or functionality. Yes, they are more work to set up, but relatively simple to manage and maintain.

For some challenges, such as authorization, Hadoop offers partial solutions. For example, user authorization for services and data access is built into HDFS. While this provides basic authorization services, it does not provide granular authorization mapping, it's not integrated with traditional authorization tools, and it really requires the application developer to design in some knowledge of owners, admins and users to fully leverage these features. Some native security features, like web proxies, are difficult to recommend given they only protect a subset of communications in the cluster.

Other threats are even harder to address as solutions will be be specific to the customer environment, or there is simply no good solution available for big data environments. API security, monitoring, granular data access, and gating MapReduce requests beyond group membership, are all issues without direct answers. In some cases, as with application security, there are simply too many variables — security controls require tradeoffs which depend heavily on specifics of the environment. Some controls, including certain types of encryption and activity monitoring, require modifications to deployments or data choke points that can slow data input to a snail's pace. In many cases we have nothing to recommend — products have not yet evolved sufficiently to handle real-world big data requirements. That does not mean you cannot fill the gaps with your own solutions. Many of the remaining operational and architectural security issues have been solved in more traditional environments, but you will need to adapt existing technologies to your environment. Regardless, the use of encryption, authentication and platform management tools will greatly improve the security of big data clusters, and close off all of the easy paths attackers use to steal information or compromise big data clusters.

We hope you find this paper useful. If you have any questions or want to discuss specifics of your situation, feel free to send us a note at info@securosis.com.

About the Author

Adrian Lane, Analyst and CTO

Adrian Lane is a Senior Security Strategist with 24 years of industry experience. He brings over a decade of C-level executive expertise to the Securosis team. Mr. Lane specializes in database architecture and data security. With extensive experience as a member of the vendor community (including positions at Ingres and Oracle), in addition to time as an IT customer in the CIO role, Adrian brings a business-oriented perspective to security implementations. Prior to joining Securosis, Adrian was CTO at database security firm IPLocks, Vice President of Engineering at Touchpoint, and CTO of the secure payment and digital rights management firm Transactor/Brodia. Adrian also blogs for Dark Reading and is a regular contributor to Information Security Magazine. Mr. Lane is a Computer Science graduate of the University of California at Berkeley with post-graduate work in operating systems at Stanford University.

About Securosis

Securosis, L.L.C. is an independent research and analysis firm dedicated to thought leadership, objectivity, and transparency. Our analysts have all held executive level positions and are dedicated to providing high-value, pragmatic advisory services.

Our services include:

- The Securosis Nexus: The Nexus is an online environment to help you get your job done better and faster. It provides pragmatic research on security topics, telling you exactly what you need to know, backed with industry-leading expert advice to answer your questions. The Nexus was designed to be fast and easy to use, and to get you the information you need as quickly as possible. Access it at <https://nexus.securosis.com/>.
- Primary research publishing: We currently release the vast majority of our research for free through our blog, and archive it in our Research Library. Most of these research documents can be sponsored for distribution on an annual basis. All published materials and presentations meet our strict objectivity requirements and conform with our Totally Transparent Research policy.
- Research products and strategic advisory services for end users: Securosis will be introducing a line of research products and inquiry-based subscription services designed to assist end user organizations in accelerating project and program success. Additional advisory projects are also available, including product selection assistance, technology and architecture strategy, education, security management evaluations, and risk assessment.
- Retainer services for vendors: Although we will accept briefings from anyone, some vendors opt for a tighter, ongoing relationship. We offer a number of flexible retainer packages. Services available as part of a retainer package include market and product analysis and strategy, technology guidance, product evaluation, and merger and acquisition assessment. We maintain our strict objectivity and confidentiality. More information on our retainer services (PDF) is available.
- External speaking and editorial: Securosis analysts frequently speak at industry events, give online presentations, and write and/or speak for a variety of publications and media.
- Other expert services: Securosis analysts are available for other services as well, including Strategic Advisory Days, Strategy Consulting Engagements, and Investor Services. These tend to be customized to meet a client's particular requirements.

Our clients range from stealth startups to some of the best known technology vendors and end users. Clients include large financial institutions, institutional investors, mid-sized enterprises, and major security vendors.

Additionally, Securosis partners with security testing labs to provide unique product evaluations that combine in-depth technical analysis with high-level product, architecture, and market analysis. For more information about Securosis, visit our website: <http://securosis.com/>.