# SecureDL: A system for securing big data

Murat Kantarcioglu [1] Fahad Shaon [2]

## 1 Introduction

As more and more data collected by organizations, managing this big data become an important challenge. To address these big data management challenges, NoSQL databases emerged as an important tool. The NoSQL (Not Only SQL) is a database mechanism developed for storage, analysis and access of large volume of structured and unstructured data. NoSQL allows schema-less data storage which is not possible with relational database systems. The benefits of using NoSQL database include high scalability, simpler designs and higher availability with more precise control. Due to these benefits, plethora of NoSQL databases have been developed in the industry. Among these NoSQL databases Hadoop and related projects (e.g., HBASE, HIVE etc.) and Spark emerged as an popular alternative due to their flexibility and integrated capabilities that allow building popular machine learning models out of the box [4].

Data Security Technologies LLC's SecureDL system enables organizations to protect their sensitive data stored in NoSQL databases such as Hadoop, Spark, HBase etc. SecureDL is a data access broker built on top of existing NoSQL databases to enforce wide range of access control, privacy and governance policies. For big data stored in NoSQL databases and data lakes, the SecureDL allows organizations to 1) enforce policies that control access to sensitive data, 2) keep necessary audit logs automatically for data governance and regulatory compliance, 3) sanitize and redact sensitive data on-the-fly based on the data sensitivity, 4) detect potentially unauthorized or anomalous access to sensitive data, 5) automatically create attribute-based access control policies based on data sensitivity and data type. [3]

Compared to existing techniques and tools available for protecting big data, the SecureDL provides advanced data sanitization, logging, intrusion detection, access control, policy generation and management capabilities in an integrated framework. SecureDL offers several important value propositions that cannot be delivered by competing technologies as we discuss in detail below.

*As big data becomes the new oil for the digital economy, we believe that realizing the* SecureDL *vision is crucial for addressing emerging pain points for securely managing big data needed for many critical applications.*

[1] murat@datasectech.com

[2] fahad@datasectech.com

[3] Our SecureDL can be extended to work with many other existing NoSQL databases. In this version of SecureDL, we focused on the two of the most popular NoSQL databases: Hadoop and Spark.

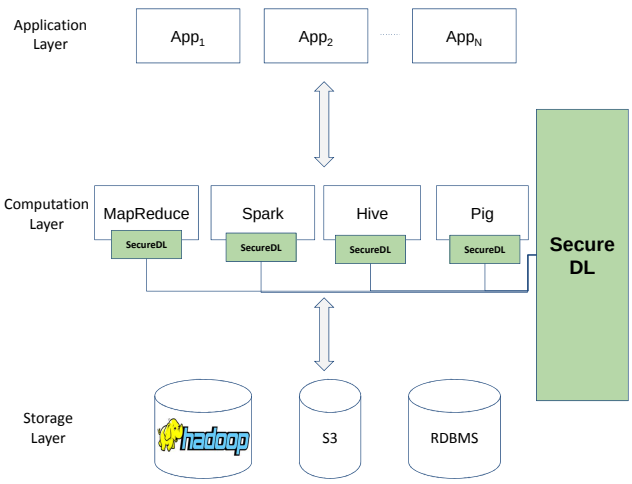


Figure 1: SecureDL Architecture Overview

## 2 Overview of SecureDL Architecture

Our SecureDL system (See Figure 1) is build on top of existing NoSQL databases such as Hadoop and Spark and designed as a data access broker where each request submitted by a user app is automatically captured by our system. These requests are logged, analyzed and then modified (if needed) to conform with security and privacy policies, and submitted to underlying NoSQL database. SecureDL is totally transparent from the user point of view and does not require any change to the user's code and/or the underlying NoSQL database systems. Therefore, it can be deployed on existing NoSQL databases.

SecureDL will enforce policies defined by the security officers before the data is accessed by a given app (See Figure 1 for more details). To enforce policies, we use aspect oriented programming technology [5] to intercept app data access requests (e.g., an app developed by data scientist Jane may try to access customer credit information for building credit score prediction module. When such an access request occurs, SecureDL observes and modifies the request to automatically remove credit card information.). Furthermore, by leveraging the machine learning models built using past access history, SecureDL can determine whether the current requests are substantially different compared to past requests. This could be used to detect potential cyber attacks including misuse by an insider assuming that cyber attacks have substantially different data access patterns compared to regular data access requests.

## 3 Overview of SecureDL Capabilities

To protect the sensitive data stored in NoSQL databases and data lakes, SecureDL intercepts each user data access/analysis request. More specifically, as shown in Figure 2, submitted request $R$ is first analyzed using the intrusion detection system by leveraging the data sensitivity of the underlying data. Unfortunately, as we discuss in more details below, this static analysis of the submitted job will be only possible if the submitted request is formed using a querying language

such as HiveQL [4]. If such a querying language is used, we can estimate the amount of sensitive data that is being accessed and apply various anomaly detection/intrusion detection mechanisms using these estimations. In addition, if a specific querying language such as HiveQL is used, our system can automatically re-write the query so that any policy will be enforced automatically. For example, we can insert a user defined function that redacts the first 5 digits of a given social security number (ssn) to a HiveQL query.

Unfortunately, query re-writing will not be possible for settings where user submits an arbitrary program. In Hadoop and Spark *a user can submit an arbitrary java, python, scala code for data analysis such as building machine learning models.* In this scenario, *we may not even know the data sets the submitted job will try to access in advance by just looking at the submitted job.* Luckily, in this case, the submitted job is converted to java byte code that eventually executed on the java virtual machines running on the nodes of a cluster [5]. Using Aspect oriented programming technique [5], we inject policy enforcement, and intrusion detection related code into the submitted user job as shown in the bottom part of Figure 2. Now modified user job will be executed using the compute layer of the NoSQL database. For example, in the case of Spark, the modified job will be executed by the Spark, and our injected policy code will determine the data sources accessed at run time and pull the cached policies as needed. Furthermore, information collected by the injected code can be used for fine grained log generation and dynamic intrusion detection.

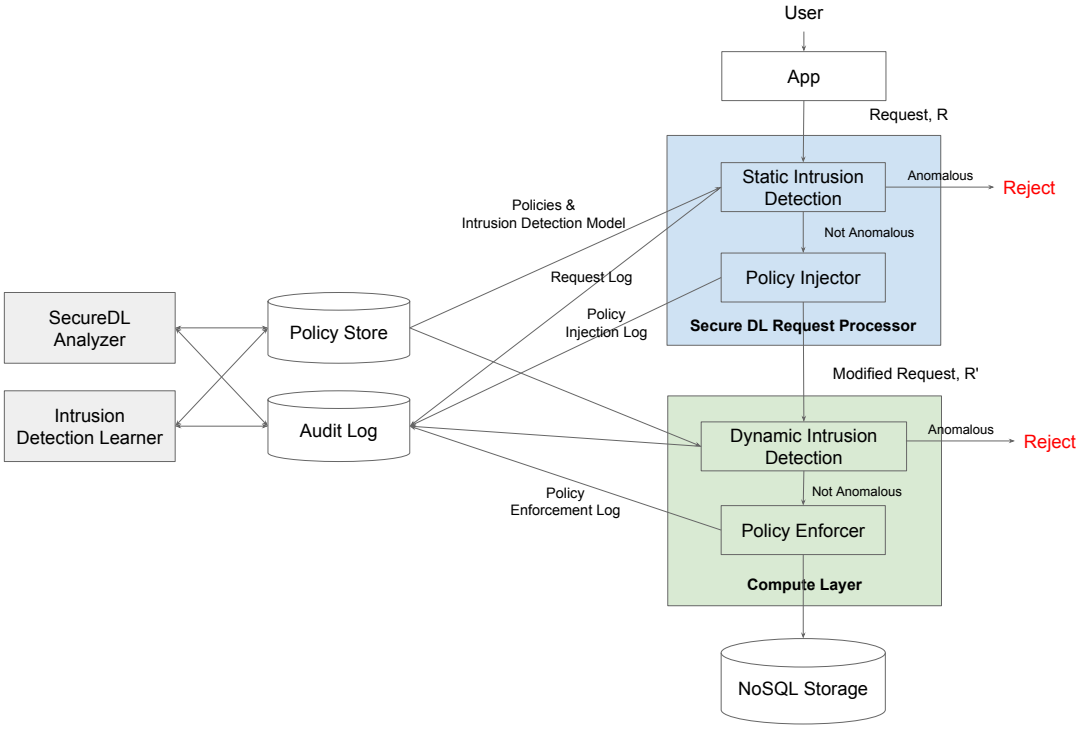


Figure 2: SecureDL Detailed Architecture

[4]HiveQL is a SQL (structured querying language) like language for querying relational data stored on Hive.
[5]A cluster may consist of hundreds or thousands of off-the-shelf computers.

### 3.1 Threat Model

We assume that all the requests are passed through our SecureDL system, and *attacker cannot directly access the data sources* via other means. Since the jobs may include arbitrary code and data, a malicious user may try to place a malicious code to circumvent the injected policies. To address this challenge, there are numerous studies (e.g.,[9]) in the literature to circumscribe untrusted codes and programs. In our case, we leverage these kinds of existing techniques (e.g., Java Security Sandbox) to prevent any access to the data other than the underlying NoSQL database specified data access interface. Furthermore, restrictions will be put in place at the java virtual machine level to prevent other potentially malicious libraries such as reflection to be used by the submitted job.

## 4 Comparison to Existing Tools

Since SecureDL addresses important pain points, we started seeing some products that try to provide some basic security, privacy and compliance policies enforcement capabilities for addressing these pain points.

Currently, there are many open source Hadoop security/privacy projects such as Apache Sentry [8], Apache Ranger [7], Apache Knox [6], Apache Atlas [1], and commercial offerings such as Cloudera Navigator [2] that provide basic security capabilities for big data.

For example, *Apache Knox* [6] provides valuable functionality that aids in the control, integration, monitoring and automation of critical administrative and analytical needs of the enterprise. Our SecureDL is actually integrated with Knox already.

In addition to these open source projects, there are some commercial offerings in the market. To our knowledge, *none of the existing products offered by competing companies have the complete capabilities that are offered by* SecureDL *system.*

Unlike these previous systems, our proposed methods can handle the ad-hoc MapReduce/Spark jobs and complex input data formats (e.g., relational, text, json etc.), and support many different types of security, privacy and governance policies. Further more, the flexible architecture of our proposed SecureDL *system that supports attribute based access control will allow us add new policies to comply with new regulations such as European Union GDPR* [3] [6]. Finally, to our knowledge, none of these existing big data security systems tries to leverage the fine grain log information, data type and data sensitivity to detect intrusions and provide comprehensive, automatic audit and data provenance generation. In summary, from technology point of view, SecureDL system is a far superior offering compared to existing alternatives for addressing important big data security and privacy pain points.

## References

[1] Atlas. Apache knox. `https://atlas.apache.org/`, 2015. Accessed: 2016-06-14.

[2] Cloudera. Cloudera navigator. `https://www.cloudera.com/products/cloudera-navigator.html/`, 2016. Accessed: 2016-06-14.

[6] please see contact us for further details.

[3] E. Commission. Eu general data protection regulation. http://ec.europa.eu/justice/data-protection/, 2016. Accessed: 2016-06-14.

[4] D. Harris. Survey shows huge popularity spike for apache spark. http://fortune.com/2015/09/25/apache-spark-survey/. 2015.

[5] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. marc Loingtier, and J. Irwin. Aspect-oriented programming. In *ECOOP*. SpringerVerlag, 1997.

[6] Knox. Apache knox. https://knox.apache.org/, 2014. Accessed: 2015-08-24.

[7] Ranger. Apache ranger. http://ranger.incubator.apache.org/, 2014. Accessed: 2015-08-24.

[8] Sentry. Apache sentry. http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/sentry.html, 2014. Accessed: 2014-05-08.

[9] M. Sun, G. Tan, J. Siefers, B. Zeng, and G. Morrisett. Bringing java's wild native world under control. *ACM Trans. Inf. Syst. Secur.*, 2013.