Securing Data in Hadoop at Uber

Mohammad Islam

Wei Han

Speaker Intro

- Mohammad Islam
  - Staff Engineer @Uber
  - Apache Hadoop contributor, PMC in Oozie & TEZ
  - Co-Authored O'Reilly book about Apache Oozie
- Wei Han
  - Technical Manager @ Uber
  - Lead Hadoop Security team

What is (NOT) covered?

- Securing Hadoop data lake at Uber
- Focus on technologies
  - Open source + internal tools

- NOT covering all aspects of data security
- NOT a **legal** advice or guidance

Data Security in Hadoop

What is Data Security?

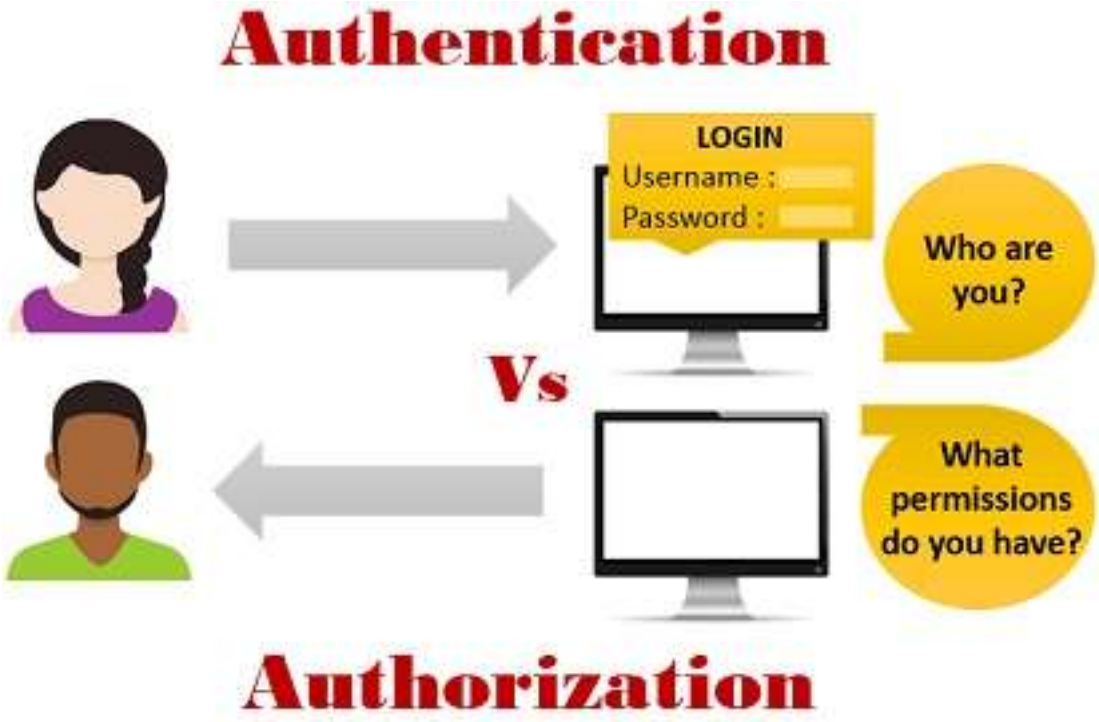
- Prevent unauthorized access to data.
- Technical focus area in data lake:
  AAAA

  Authentication
  Authorization
  Auditing
  Anonymization

4 Pillars of Data Security (1/2)

1. Authentication (AuthN)
Verify identity of a user

2. Authorization (AuthZ)
Access control of data



4 Pillars of Data Security (2/2)

3. Auditing
- Post-mortem
- Anomaly detection

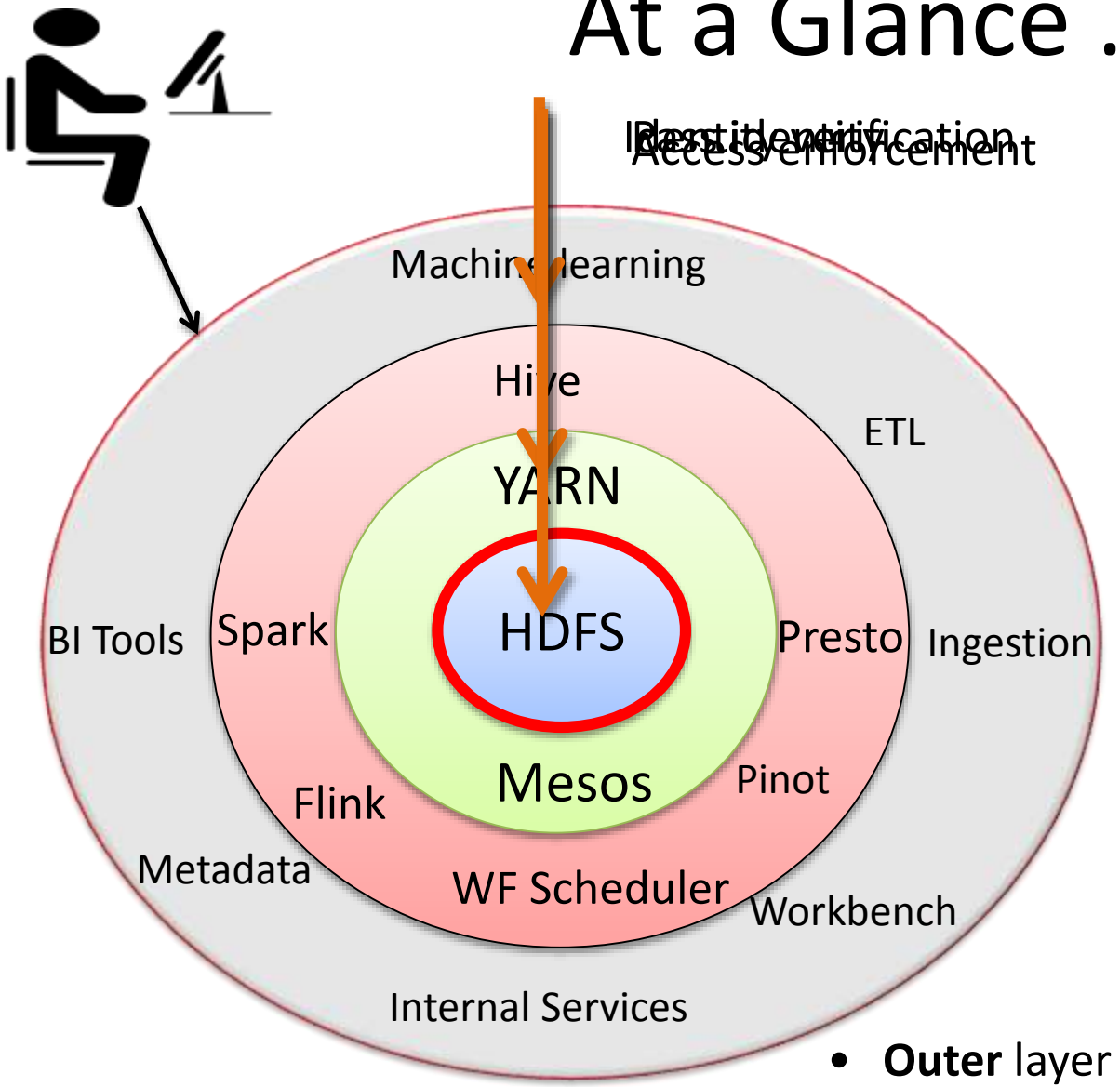
4. Anonymization
- tokenization
- Masking etc.

Design Considerations

- Secure **all** access paths to HDFS
- Enforcement at the lowest-level
- User/group based (AD) access control
- Centralized policy store

Not at the cost of infrastructure flexibility

At a Glance ..



- **Outer** layer verifies user identity
- **Middle** layers securely pass the iden
- **Innermost** layer enforces access cont

Authentication

Authentication Overview

- Kerber and delegation token (DT) for Hadoop
- Custo S2S authN for non-Hadoop Services

AuthN Protocol Translation - Knox

- Why?
  - Seamless integration among AuthN protocols
    - Translate custom AuthN protocols to Kerberos
- Contributed to Apache Knox
  - Pluggable AuthN validator for any custom AuthN protocol (KNOX-861, KNOX-869)
  - Improved monitoring (KNOX-940)

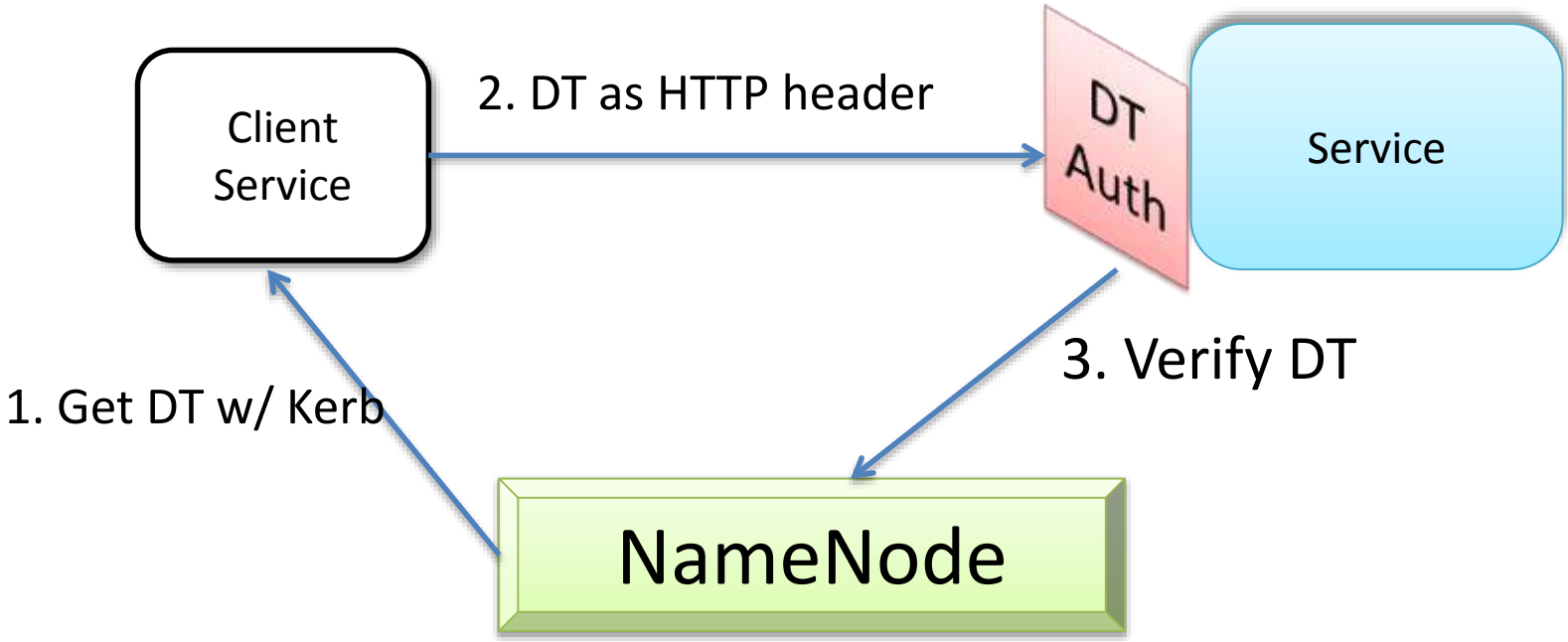
Impersonation/Delegation

- Why?
  - Hadoop already supports impersonation or doAs
    - Work *on-behalf-of* others
  - Internal authN mechanism doesn't support it
- How?
  - Utilize Apache Knox
  - Whitelist the impersonated services using config
  - Idea borrowed from Hadoop core-site.xml

Off-label Usage : Delegation Token

- Delegation token (DT) is a Hadoop concept
- Used DT for authentication *only* when:
  - Other protocol doesn't work or is not ready
  - DT for HDFS is already available
  - HTTP REST service
- Added support in Presto and few other internal services

Off-label Usage : Delegation Token :

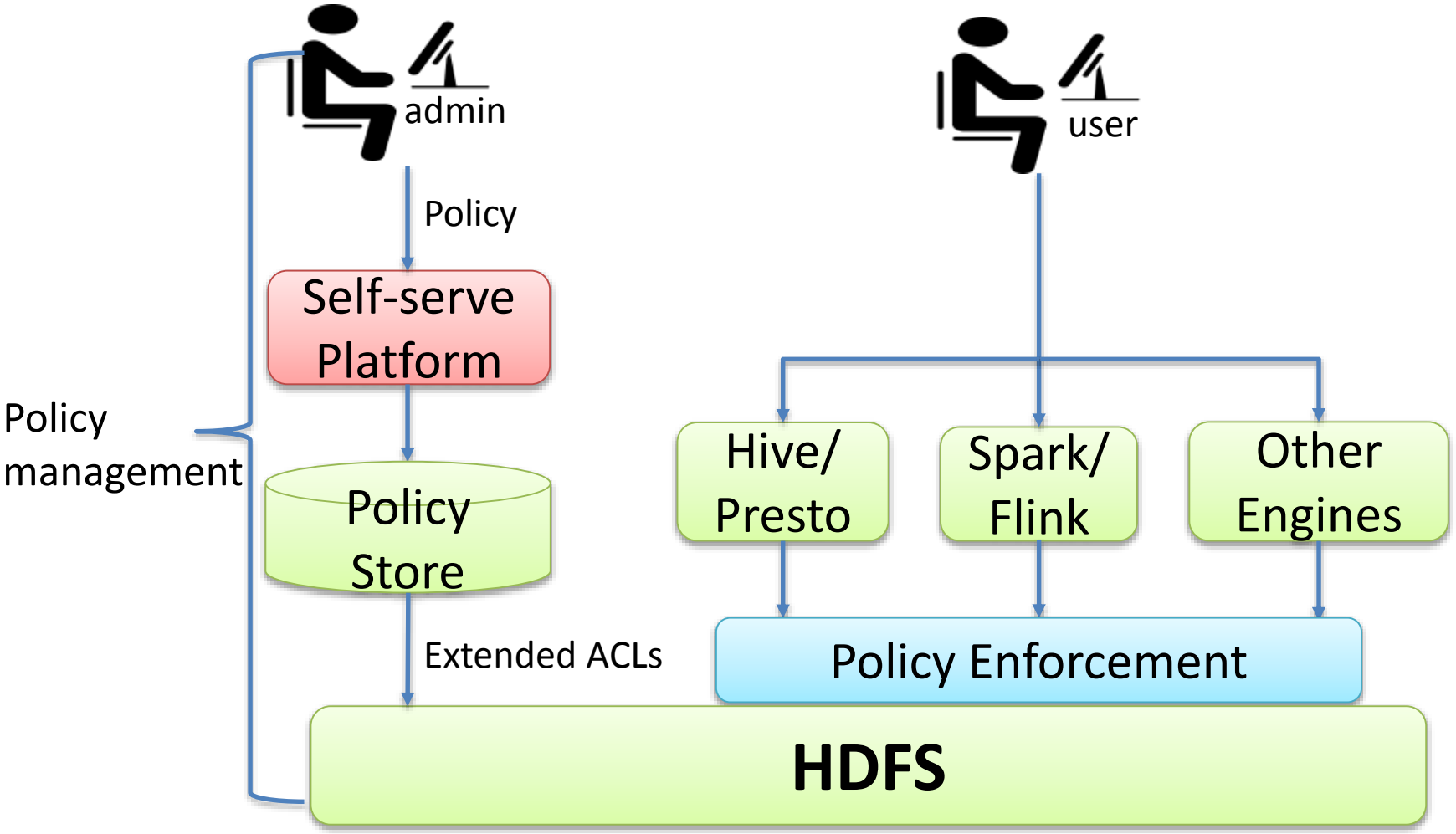


Summary

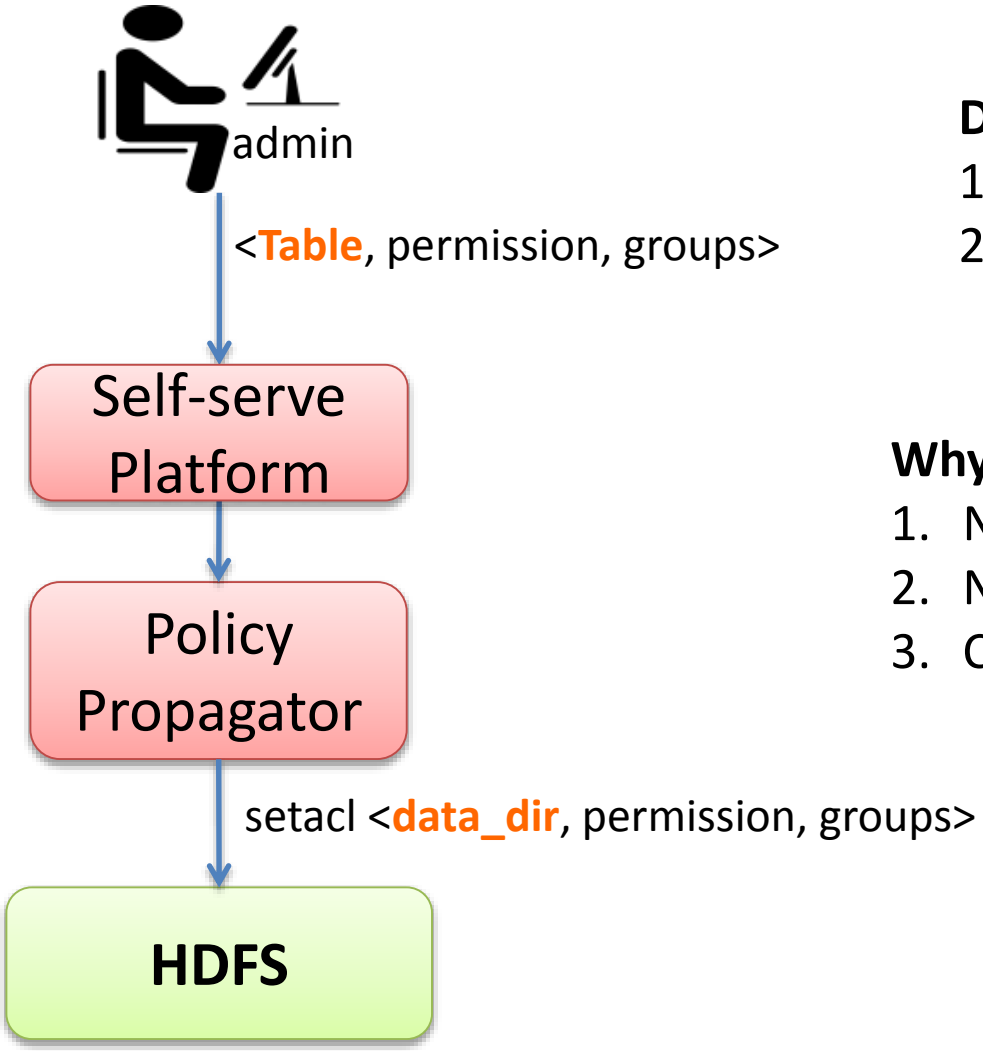
- (+) Quick and easy to implement
- (-) Extra load on NN (caching can address it)

Authorization

Authorization



RBAC Policy to ACLs



Design Options

1. Use HDFS plugin (Sentry/Ranger)
2. Set the ACLs **directly** in HDFS

Why Option 2?

1. No overhead (Mem/latency) to NN
2. NN Scale/stability is a concern
3. Challenge : Keep in sync

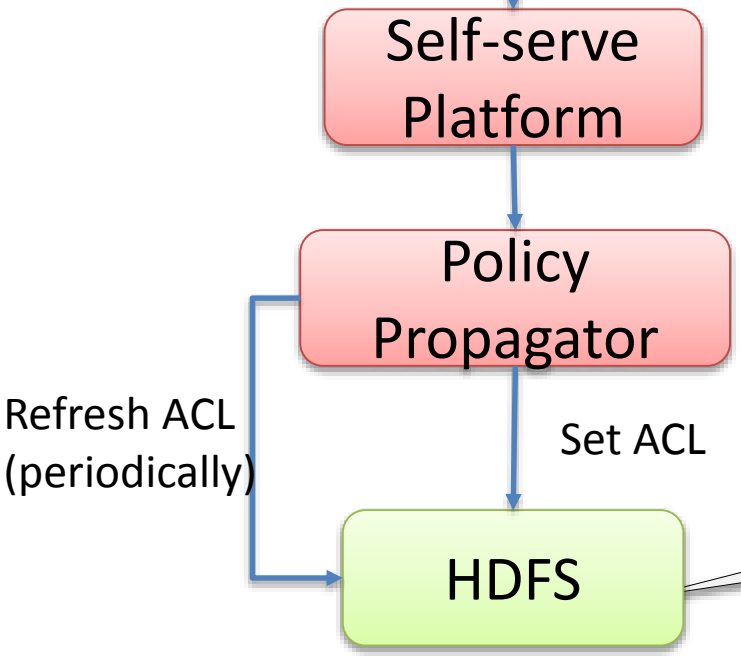
Partition-based Access Control (1/2)

- Policy defined at partition level
  - Usual access control is table-level
  - Access can change based on **time or geography**
- Example use case:
  - "*events*" table is partitioned by date
  - Policies
    - By default, employee can only access new events records
    - Only authorized groups can access events records older than X days

Partition-based Access Control (2/2)

Sample Policies

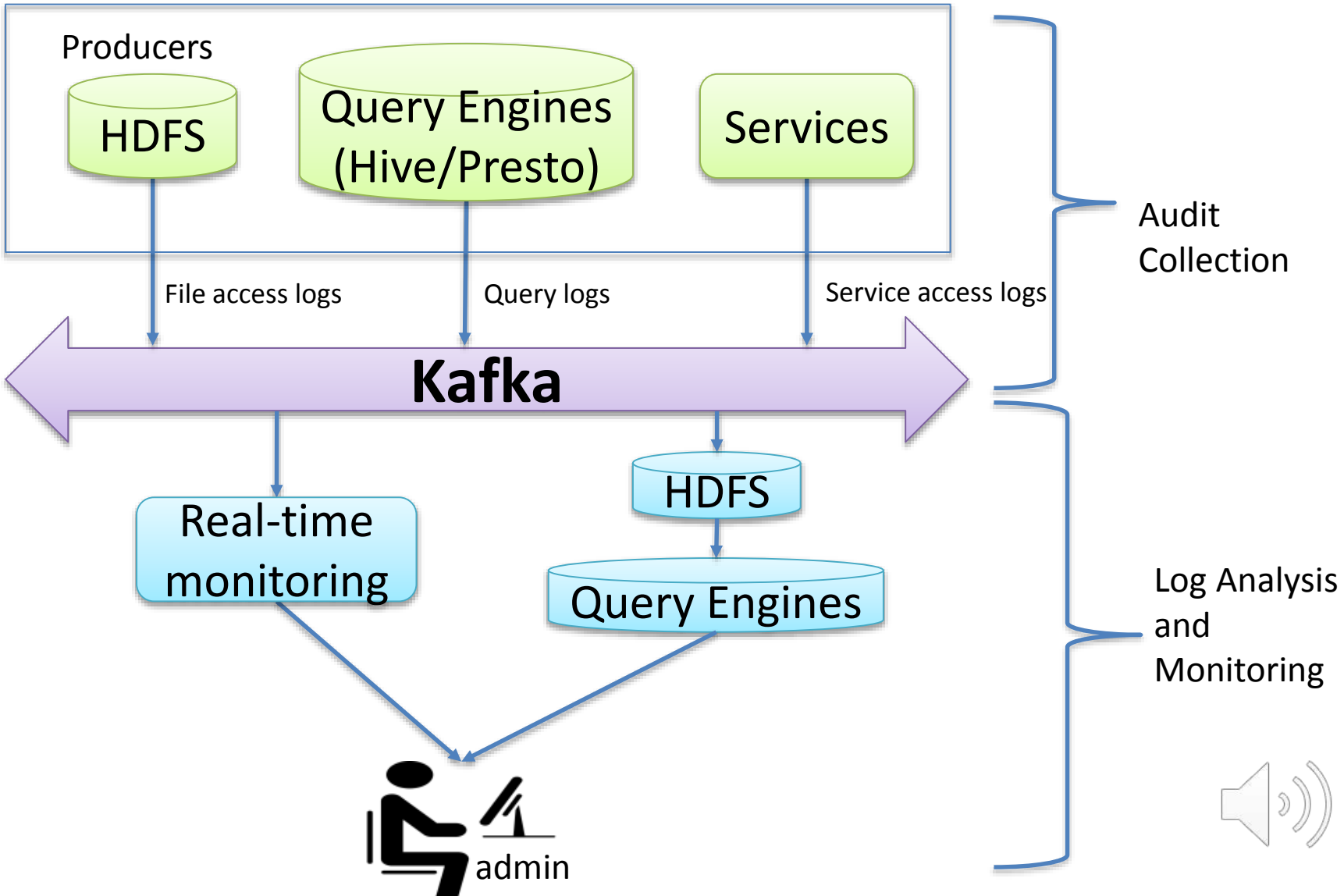
Table	Privilege	Group	**Time restriction**
events	read	**employee**	**X days**
events	read	authorized_group	**none**



/events/date=2018.06.06
group: **employee**, authorized_group:**r-x**
/events/date=2017.02.06
group: authorized_group:**r-x**

Auditing

Auditing


Producers
HDFS
Query Engines (Hive/Presto)
Services
File access logs
Query logs
Service access logs
Kafka
Audit Collection
Real-time monitoring
HDFS
Query Engines
Log Analysis and Monitoring
admin

Anonymization

Anonymization

- Transform any data into **unidentifiable** form
- Loosely used to mention:
  - Removal
  - Redaction
  - Masking
  - Tokenization

Next : Enforce AuthZ through Encryption

Column-level Access Control

- Why?
  - When only some columns in a table are sensitive and need special access control
  - Finer grained access control based on level of sensitivity

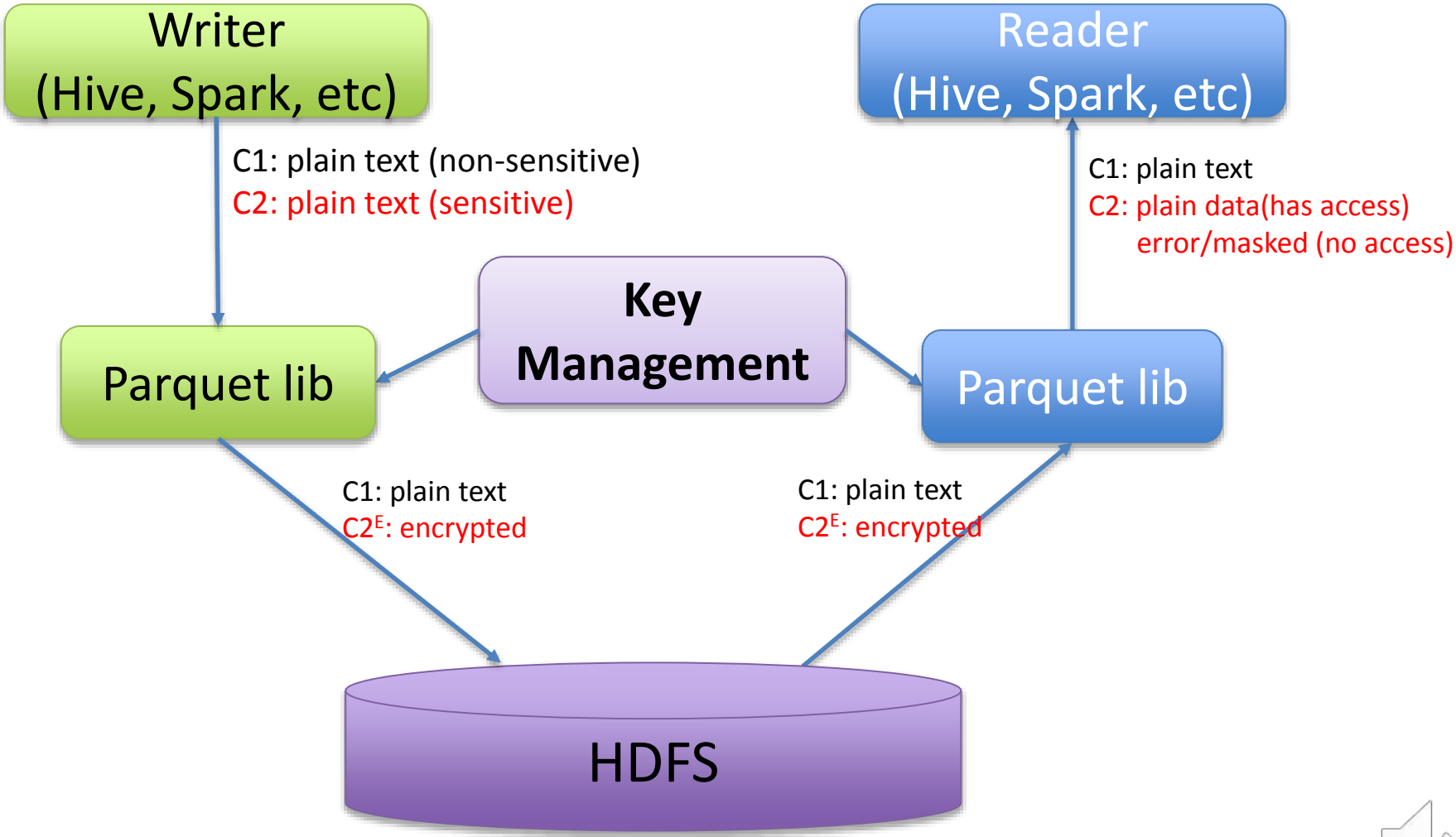
Column	C_1	C_2	C_3	….	C_{15}	…	C_{34}	…	C_{100}
Sensitivity Level	0	8	0	0	6	0	9	0	0

- Challenges
  - Enforce on common access paths
  - HDFS doesn't understand column

Approach

- Enforce in data format (**Parquet**) level
- **Encrypt** *only* sensitive columns in HDFS
- Access controlled through encryption key management
- Different column can have different key for encryption/decryption
- Open source activities: PARQUET-1178 And PARQUET-1325

Column-level Access Control



Conclusion

Take Away

1. Security scope within Hadoop is expanding
   - Conventional thinking is being challenged
   - Need significant changes in the architecture
2. Finer-grain security is must for big data
   - Column/Partition/Row level access control
3. Security by design is critical
   - Retrofitting is very hard

Q&A