# Arenadata™ Hadoop

*Версия - v1.5.2*

## Настройка безопасности для Arenadata Hadoop

*2018 Arenadata™*

# Оглавление

**1 Функции безопасности Hadoop** **3**
1.1 Безопасность периметра . . . 3
1.2 Аутентификация . . . 3
1.3 Авторизация (контроль доступа) . . . 3
1.4 Отчетность (аудит и мониторинг безопасности) . . . 4
1.5 Защита данных . . . 4

**2 Настройка авторизации в Hadoop** **5**
2.1 Установка Ranger с помощью Ambari . . . 5
2.2 Создание политики HDFS . . . 12

В документе приведены функции безопасности Hadoop, такие как безопасность периметра, аутентификация, авторизация, отчетность и защита данных.

Документ может быть полезен администраторам, программистам, разработчикам и сотрудникам подразделений информационных технологий, осуществляющих внедрение кластера.

**Important:** Контактная информация службы поддержки – e-mail: info@arenadata.io

# Глава 1

# Функции безопасности Hadoop

Для организаций, которые хранят конфиденциальные данные в экосистеме **Hadoop**, например, запатентованные или персональные данные, которые подлежат соблюдению нормативных требований (**HIPPA**, **PCI**, **DSS**, **FISAM** и т.д.), безопасность очень важна. Многие организации также должны придерживаться строгой внутренней политики безопасности.

**Arenadata Hadoop** (далее – **ADH**) обеспечивает комплексный подход к безопасности в следующих ключевых областях: безопасность периметра, аутентификация, авторизация, отчетность, защита данных.

## 1.1 Безопасность периметра

**ADH** позволяет изолировать кластер **Hadoop** с помощью шлюза и правильно настроенных правил брандмауэра. **ADH** поддерживает следующую защиту периметра:

- Apache Knox Gateway;
- Клиенты шлюза.

## 1.2 Аутентификация

**ADH** предоставляет единую точку аутентификации для сервисов и пользователей, которая интегрируется с существующими системами идентификации предприятия и доступа. **ADH** поддерживает следующие службы аутентификации:

- Kerberos;
- LDAP;
- Локальная система Unix;
- SSO (по периметру через Apache Knox Gateway).

## 1.3 Авторизация (контроль доступа)

**ADH** предоставляет функции, позволяющие системным администраторам контролировать доступ к данным **Hadoop** с использованием авторизации на основе ролей. **ADH** поддерживает следующие модели авторизации:

- Высококачественное управление доступом для данных, хранящихся в HDFS;
- Контроль доступа на уровне ресурсов для YARN;
- Контроль доступа на крупнозернистом уровне для операций MapReduce;
- Контроль доступа на уровне семейства таблиц и столбцов для данных HBase;
- Контроль доступа на уровне таблицы для наборов данных Apache Hive.

## 1.4 Отчетность (аудит и мониторинг безопасности)

**ADH** позволяет отслеживать активность **Hadoop**, используя Native Auditing, журналы аудита безопасности периметра на шлюзе **Knox**, и из центрального расположения консоли администрирования **ADH**, включая:

- Запросы доступа;
- Операции обработки данных;
- Изменение данных.

## 1.5 Защита данных

**ADH** предоставляет механизмы для шифрования данных при их передаче и требует использования партнерских решений для шифрования данных в состоянии покоя, их поиска и маскирования. **ADH** поддерживает следующие методы шифрования:

- SSL для компонентов ADH;
- Шифрование RPC;
- Протокол передачи данных.

# Глава 2

# Настройка авторизации в Hadoop


- *Установка Ranger с помощью Ambari*
- *Создание политики HDFS*


## 2.1 Установка Ranger с помощью Ambari

**Apache Ranger** можно установить при помощи пользовательского интерфейса **Ambari** или вручную, используя платформу **Arenadata Hadoop**. В отличие от ручного процесса установки, требующего выполнения ряда шагов, установка **Ranger** с использованием интерфейса **Ambari** проще и легче. Опция службы **Ranger** доступна через мастер **Add Service** после инсталляции кластера **ADH** с помощью установщика.

После установки и настройки **Ambari** можно использовать мастер добавления служб для установки следующих компонентов:

- Ranger Admin
- Ranger UserSync
- Ranger Key Management Service

После установки и запуска этих компонентов можно включить плагины **Ranger**, перейдя к каждому отдельному сервису **Ranger** (**HDFS**, **HBase**, **Hiveserver2**, **Storm**, **Knox**, **YARN** и **Kafka**) и изменив конфигурацию в расширенном режиме *ranger-<service>-plugin-properties*.

**Important:** При включении плагина Ranger необходимо перезапустить компонент

**Important:** Включение Apache Storm или Apache Kafka требует включения Kerberos

При обновлении **ADH** (на **Ambari 2.5.0**) установка **Ranger DB** выполняется при первом запуске **Ranger** (в предыдущих версиях настройка **Ranger DB** выполнялась во время установки). Это означает, что **Ranger** при первом запуске может занять больше времени, чем раньше (последующие перезагрузки будут такими же быстрыми, как и раньше).

### 2.1.1 Предварительные требования к установке

Перед установкой **Ranger** необходимо убедиться, что кластер отвечает следующим требованиям:

- Рекомендуется хранить аудиты как в HDFS, так и в Solr. Конфигурация по умолчанию для Ranger Audits в Solr использует общий экземпляр Solr, предоставляемый сервисом Ambari Infra (дополнительные сведения см. в разделе Ranger Audit Settings);
- Чтобы обеспечить принудительную авторизацию на уровне групп LDAP/AD в Hadoop, необходимо настроить сопоставление групп Hadoop для LDAP/AD для LDAP (*Настройка сопоставления групп Hadoop для LDAP/AD*);
- Должен быть запущен и доступен экземпляр базы данных MySQL, Oracle, PostgreSQL или Amazon RDS, который будет использоваться Ranger. Установщик Ranger создаст двух новых пользователей (имена по умолчанию: *rangeradmin* и *rangerlogger*) и две новые базы данных (имена по умолчанию: *ranger* и *ranger_audit*).

  Конфигурация экземпляра для Ranger для некоторых баз данных описана в следующих разделах:

  - *Конфигурация MySQL*
  - *Конфигурация PostgreSQL*
  - *Конфигурация Oracle*

  При использовании Amazon RDS есть дополнительные требования (*Требования к Amazon RDS*).
- При решении не предоставлять данные учетной записи администратора базы данных (DBA) установщику Ambari Ranger, можно использовать Python-скрипт *dba_script.py* для создания пользователей базы данных Ranger DB без предоставления этой информации установщику. После чего запустить обычную установку Ambari Ranger без указания имени и пароля администратора. Дополнительные сведения приведены в разделе *Настройка пользователей базы данных без совместного использования учетных данных DBA*.

### Настройка сопоставления групп Hadoop для LDAP/AD

Для обеспечения принудительной авторизации на уровне групп **LDAP/AD** в **Hadoop**, необходимо настроить сопоставление групп **Hadoop** для **LDAP/AD**.

---

**Important:** Доступ к LDAP и сведения о подключении: настройки LDAP могут различаться в зависимости от используемой реализации LDAP

---

Существует три способа настройки сопоставления групп **Hadoop**.

- **Настройка сопоставления групп Hadoop для LDAP/AD с использованием SSSD (рекомендуется)**

Для сопоставления групп рекомендуется использовать SSSD или один из следующих сервисов подключения ОС **Linux** к **LDAP**:

- Centrify
- NSLCD
- Winbind
- SAMBA

Большинство перечисленных сервисов позволяет не только искать пользователя и перечислять группы, но также выполнять другие действия на хосте. При этом ни одно из этих действий не требуется для сопоставления групп **LDAP** в **Hadoop**. Поэтому, оценивая эти сервисы, необходимо понимать разницу между модулем **NSS** (который выполняет разрешение пользователь/группа) и модулем **PAM** (который выполняет аутентификацию пользователя). Для возможности поиска (или “валидации”) пользователя в **LDAP** и перечисления групп требуется **NSS**. А **PAM** может представлять угрозу безопасности.

- **Настройка сопоставления групп Hadoop в файле core-site.xml**

Настройка **Hadoop** для использования сопоставления групп на основе **LDAP** в файле *core-site.xml* осуществляется в следующем порядке:

1. Добавить свойства, показанные в приведенном ниже примере, в файл *core-site.xml*. Необходимо указать значение для привязанного пользователя, его пароль и другие свойства, специфичные для экземпляра LDAP, и убедиться, что фильтры классов объектов, пользователей и групп соответствуют значениям, указанным в экземпляре LDAP.

```
<property>
<name>hadoop.security.group.mapping</name>
<value>org.apache.hadoop.security.LdapGroupsMapping</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.bind.user</name>
<value>cn=Manager,dc=hadoop,dc=apache,dc=org</value>
</property>

<!-
<property>
<name>hadoop.security.group.mapping.ldap.bind.password.file</name>
<value>/etc/hadoop/conf/ldap-conn-pass.txt</value>
</property>
->

<property>
<name>hadoop.security.group.mapping.ldap.bind.password</name>
<value>hadoop</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.url</name>
<value>ldap://localhost:389/dc=hadoop,dc=apache,dc=org</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.url</name>
<value>ldap://localhost:389/dc=hadoop,dc=apache,dc=org</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.base</name>
<value></value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.filter.user</name>
<value>(&amp;(|(objectclass=person)(objectclass=applicationProcess))(cn={0}))</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.filter.group</name>
<value>(objectclass=groupOfNames)</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.attr.member</name>
```

```
<value>member</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.attr.group.name</name>
<value>cn</value>
</property>
```

2. В зависимости от конфигурации можно обновлять сопоставления пользователей и групп с помощью следующих команд HDFS и YARN:

```
hdfs dfsadmin -refreshUserToGroupsMappings
yarn rmadmin -refreshUserToGroupsMappings
```

3. Проверить сопоставление групп LDAP, выполнив команду *hdfs groups*. Команда отображает группы из LDAP для текущего пользователя. При настроенном сопоставлении групп LDAP разрешения HDFS могут использовать группы, определенные в LDAP для контроля доступа.

- **Ручное создание пользователей и групп в среде Linux**

  Также можно вручную создавать пользователей и группы в среде Linux.

### Конфигурация MySQL

При использовании **MySQL** машина для хранения таблиц политики администратора **Ranger** обязательно должна поддерживать транзакции. **InnoDB** – это пример машины, поддерживающей транзакции.

При использовании **Amazon RDS** есть дополнительные требования (*Требования к Amazon RDS*).

Для конфигурации экземпляра для **Ranger** для **MySQL** необходимо выполнить следующие шаги:

1. Для создания баз данных Ranger должен использоваться администратор базы данных MySQL. Для создания пользователя *rangerdba* с паролем *rangerdba* необходимо:

- Войти в систему как пользователь *root* и использовать следующие команды, чтобы создать пользователя *rangerdba* и предоставить ему соответствующие права:

```
CREATE USER 'rangerdba'@'localhost' IDENTIFIED BY 'rangerdba';

GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'localhost';

CREATE USER 'rangerdba'@'%' IDENTIFIED BY 'rangerdba';

GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'%';

GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'localhost' WITH GRANT OPTION;

GRANT ALL PRIVILEGES ON *.* TO 'rangerdba'@'%' WITH GRANT OPTION;

FLUSH PRIVILEGES;
```

- Использовать команду *exit* для выхода из MySQL;
- Теперь можно подключиться к базе данных как *rangerdba*, используя следующую команду:

```
mysql -u rangerdba -prangerdba
```

После тестирования входа в систему *rangerdba* использовать команду *exit* для выхода из MySQL.

2. Следующая команда используется для подтверждения, что файл *mysql-connector-java.jar* находится в папке общего доступа Java. Команда должна быть запущена на сервере, на котором установлен сервер Ambari:

```
ls /usr/share/java/mysql-connector-java.jar
```

Если файл находится не в каталоге общего доступа Java, использовать следующую команду для установки соединения:

- RHEL/CentOS/Oracle Linux:

```
yum install mysql-connector-java*
```

- SLES:

```
zypper install mysql-connector-java*
```

3. Использовать следующий формат команды, чтобы установить путь *jdbc/driver/path* на основе местоположения файла *.jar* драйвера MySQL JDBC. Команда должна выполняться на сервере, на котором установлен сервер Ambari:

```
ambari-server setup --jdbc-db={database-type} --jdbc-driver={/jdbc/driver/path}
```

Например:

```
ambari-server setup --jdbc-db=mysql --jdbc-driver=/usr/share/java/mysql-connector-java.jar
```

### Конфигурация PostgreSQL

При использовании **Amazon RDS** есть дополнительные требования (*Требования к Amazon RDS*).

Для конфигурации экземпляра для **Ranger** для **PostgreSQL** необходимо выполнить следующие шаги:

1. На хосте PostgreSQL установить соответствующий коннектор PostgreSQL:

- RHEL/CentOS/Oracle Linux:

```
yum install postgresql-jdbc*
```

- SLES:

```
zypper install -y postgresql-jdbc
```

2. Убедиться, что файл *.jar* находится в папке общего доступа Java:

```
ls /usr/share/java/postgresql-jdbc.jar
```

3. Изменить режим доступа файла *.jar* на *644*:

```
chmod 644 /usr/share/java/postgresql-jdbc.jar
```

4. Для создания баз данных Ranger должен использоваться администратор базы данных PostgreSQL. Для создания пользователя *rangerdba* и предоставления ему соответствующих прав следует использовать команду:

```
echo "CREATE DATABASE $dbname;" | sudo -u $postgres psql -U postgres
echo "CREATE USER $rangerdba WITH PASSWORD '$passwd';" | sudo -u $postgres psql -U postgres
echo "GRANT ALL PRIVILEGES ON DATABASE $dbname TO $rangerdba;" | sudo -u $postgres psql -U␣
↪postgres
```

Где *$postgres* – пользователь Postgres, *$dbname* – имя базы данных PostgreSQL.

5. Использовать следующий формат команды, чтобы установить путь *jdbc/driver/path* на основе местоположения файла *.jar* драйвера PostgreSQL JDBC. Команда должна выполняться на сервере, на котором установлен сервер Ambari:

```
ambari-server setup --jdbc-db={database-type} --jdbc-driver={/jdbc/driver/path}
```

Например:

```
ambari-server setup --jdbc-db=postgres --jdbc-driver=/usr/share/java/postgresql-jdbc.jar
```

6. Выполнить следующую команду:

```
export HADOOP_CLASSPATH=${HADOOP_CLASSPATH}:${JAVA_JDBC_LIBS}:/connector jar path
```

7. Разрешить доступ *Allow Access* для пользователей Ranger:

- изменить *listen_ addresses='localhost'* на *listen_ addresses=''* ('*' = any)*, чтобы прослушивать все IP-адреса в *postgresql.conf*;
- внести следующие изменения пользователям *Ranger db* и *Ranger audit db* в файле *pg_ hba.conf* (Рис.2.1.).

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD

# "local" is for Unix domain socket connections only
local   all     postgres,rangeradmin,rangerlogger       trust
# IPv4 local connections:
host    all     postgres,rangeradmin,rangerlogger       0.0.0.0/0          trust
# IPv6 local connections:
host    all     postgres,rangeradmin,rangerlogger       ::/0               trust
"/var/lib/pgsql/data/pg_hba.conf" 74L, 3445C
```

Рис.2.1.: Необходимые изменения пользователям Ranger db и Ranger audit db

8. После редактирования файла *pg_ hba.conf* запустить команду для обновления конфигурации базы данных PostgreSQL:

```
sudo -u postgres /usr/bin/pg_ctl -D $PGDATA reload
```

Например, если файл *pg_ hba.conf* находится в каталоге */var/lib/pgsql/data*, значением *$PGDATA* является */var/lib/pgsql/data*.

### Конфигурация Oracle

При использовании **Amazon RDS** есть дополнительные требования (*Требования к Amazon RDS*).

Для конфигурации экземпляра для **Ranger** для **Oracle** необходимо выполнить следующие шаги:

1. На узле Oracle установить соответствующий JDBC-файл *.jar*:

- Загрузить драйвер Oracle JDBC (OJDBC)
- Для Oracle Database 11g: выбрать Oracle Database 11g Release 2 drivers > ojdbc6.jar
- Для Oracle Database 12c: выбрать Oracle Database 12c Release 1 driver > ojdbc7.jar
- Скопировать файл *.jar* в папку общего доступа Java. Например, *cp ojdbc7.jar /usr/share/java/*

- Убедиться, что .jar-файл имеет соответствующие разрешения:

```
chmod 644 /usr/share/java/ojdbc7.jar
```

2. Для создания баз данных Ranger должен использоваться администратор базы данных Oracle.

Для создания пользователя *RANGERDBA* и предоставления ему прав с помощью SQL*Plus – утилиты администрирования базы данных Oracle, следует использовать команду:

```
# sqlplus sys/root as sysdba
CREATE USER $RANGERDBA IDENTIFIED BY $RANGERDBAPASSWORD;
GRANT SELECT_CATALOG_ROLE TO $RANGERDBA;
GRANT CONNECT, RESOURCE TO $RANGERDBA;
QUIT;
```

3. Использовать следующий формат команды, чтобы установить путь *jdbc/driver/path* на основе местоположения файла *.jar* драйвера Oracle JDBC. Команда должна выполняться на сервере, на котором установлен сервер Ambari:

```
ambari-server setup --jdbc-db={database-type} --jdbc-driver={/jdbc/driver/path}
```

Например:

```
ambari-server setup --jdbc-db=oracle --jdbc-driver=/usr/share/java/ojdbc6.jar
```

### Требования к Amazon RDS

**Ranger** требует наличия реляционной базы данных в качестве хранилища политик. Существуют дополнительные требования для баз данных на основе **Amazon RDS** из-за специфичности настроек и управления.

- **MySQL/MariaDB**

Во время установки **Ranger** необходимо изменить переменную *log_bin_trust_function_creators* на значение *1*. Через панель управления RDS Dashboard > Parameter group (в левой части страницы):

- Установить переменную MySQL Server *log_bin_trust_function_creators* в значение *1*.
- (Опционально) после завершения установки Ranger сбросить значение параметра *log_bin_trust_function_creators* в исходное значение (требование к значению переменной относится только на время установки Ranger).

Дополнительная информация:

- Stratalux: Why You Should Always Use a Custom DB Parameter Group When Creating an RDS Instance
- AWS Documentation>Amazon RDS DB Instance Lifecycle » Working with DB Parameter Groups
- MySQL 5.7 Reference Manual >Binary Logging of Stored Programs

- **PostgreSQL**

Пользователь базы данных **Ranger** на сервере **Amazon RDS PostgreSQL Server** должен быть создан до установки **Ranger** и ему должна быть предоставлена роль *CREATEDB*.

1. Используя основную учетную запись пользователя (заведенную при создании экземпляра RDS PostgreSQL), войти в Amazon RDS PostgreSQL Server и выполнить команды:

```
CREATE USER $rangerdbuser WITH LOGIN PASSWORD 'password'

GRANT $rangerdbuser to $postgresroot
```

```
```

Где *$postgresroot* – это основная учетная запись пользователя RDS PostgreSQL (например, *postgresroot*), а *$rangerdbuser* – имя пользователя базы данных Ranger (например: *rangeradmin*).

2. Если используется Ranger KMS, выполнить следующие команды:

```
CREATE USER $rangerkmsuser WITH LOGIN PASSWORD 'password'

GRANT $rangerkmsuser to $postgresroot
```

Где *$postgresroot* – это основная учетная запись пользователя RDS PostgreSQL (например, *postgresroot*), а *$rangerkmsuser* – имя пользователя Ranger KMS (например, *rangerkms*).

- **Oracle**

Из-за ограничений в Amazon RDS создание пользователя базы данных **Ranger** и табличного пространства, а так же предоставление пользователю **Ranger** необходимых привилегий выполняется вручную.

1. Используя основную учетную запись пользователя (заведенную при создании экземпляра RDS Oracle), войти в RDS Oracle Server и выполнить команды:

```
create user $rangerdbuser identified by “password”;
GRANT CREATE SESSION,CREATE PROCEDURE,CREATE TABLE,CREATE VIEW,CREATE SEQUENCE,CREATE PUBLIC␣
↪SYNONYM,CREATE ANY SYNONYM,CREATE TRIGGER,UNLIMITED Tablespace TO $rangerdbuser;
create tablespace $rangerdb datafile size 10M autoextend on;
alter user $rangerdbuser DEFAULT Tablespace $rangerdb;
```

Где *$rangerdb* – это фактическое имя базы данных Ranger (например, *ranger*), а *$rangerdbuser* – имя пользователя Ranger (например: *rangeradmin*).

2. Если используется Ranger KMS, выполнить следующие команды:

```
create user $rangerdbuser identified by “password”;
GRANT CREATE SESSION,CREATE PROCEDURE,CREATE TABLE,CREATE VIEW,CREATE SEQUENCE,CREATE PUBLIC␣
↪SYNONYM,CREATE ANY SYNONYM,CREATE TRIGGER,UNLIMITED Tablespace TO $rangerkmsuser;
create tablespace $rangerkmsdb datafile size 10M autoextend on;
alter user $rangerkmsuser DEFAULT Tablespace $rangerkmsdb;
```

Где *$rangerkmsdb* – это фактическое имя базы данных Ranger (например: *rangerkms*), а *$rangerkmsuser* – имя пользователя Ranger (например: *rangerkms*).

#### Настройка пользователей базы данных без совместного использования учетных данных DBA

## 2.2 Создание политики HDFS