IBM® SECURITY GUARDIUM® ACTIVITY MONITORING (AND BLOCKING) FOR HORTONWORKS HADOOP USING APACHE RANGER INTEGRATION

Version 10.1.2

(updated 01.09.2017) –fixed typo for gardium_ evaluator.jar and <current version>

(updated 9/5/2017)-

- Added information about guard_log4j_config.py as an option to configure Ranger when UI or grdapis cannot be used.
- Additional cleanup and clarification around some steps (such as when service restart required)

Contents

Overview of the Guardium integration with Apache Ranger

Apache Ranger, included in the Hortonworks Data Platform, offers fine-grained access control and auditing over Hadoop components, such as Hive, HBASE, HDFS, and so on by using *policies*. The audit data is written to both HDFS and to Solr (recommended). Guardium can integrate with Ranger in two ways:

- For auditing, Guardium acts as another logger source for Ranger Auditing. Audited activity is sent to the Guardium collector where it is parsed and logged. Once the data is in Guardium, it is highly protected in the hardened appliance, and all normal Guardium functions can be used such as real time alerting and integration with SIEM, reporting and workflow, and analytics.
- For blocking, Guardium extends Ranger access control policies, using what is known in Ranger as *dynamic policies*.

Why integrate with Ranger rather than using "standard" Guardium UNIX S-TAPs for monitoring and blocking? A key reason is the fact that many organizations are now using SSL encryption from their clients to access Hadoop data. By using this integration, the data is decrypted before it is sent to the Guardium appliance for auditing.

If blocking is a requirement, integration with Ranger using dynamic policies enables blocking support for more components than is supported using standard S-TAP.

Prerequisites

The integration with Ranger requires the following minimum software release levels:

- IBM Security Guardium 10.1 (S-TAP and Appliance)
- Hortonworks 2.3 with Ranger

When to use standard S-TAP inspection engines vs S-TAP+Ranger

Although you can use both inspection engines and Ranger integration in the same cluster, it is unlikely that you would need this. The table below specifies which functions are available with each method.

Table 1. When to use Ranger integration

Desired function	Standard S-TAP	S-TAP+Ranger	Considerations
Audit SSL-encrypted activity	✗	✓	
Audit Kerberos-authenticated traffic	✓	✓	By using Ranger integration, no need to propagate Keytabs for Guardium.

Audit Hive, HBASE, HDFS	✓	✓	HBASE deployment is simpler with Ranger as there is no need to deploy S-TAPs on the region servers. See the Guardium Deployment Guide for Hadoop for details on how data is logged for various components using standard S-TAP.
Audit SOLR	✓	×	This is on the roadmap for S-TAP+Ranger. For regular S-TAP use HTTP inspection engine and computed attribute to extract user name.
Audit Kafka	×	✓	
Audit Storm	×	✓	This is new with 10.1.2. Requires Ambari 2.2 or higher.
Audit Yarn	✓	×	The Ranger integration *does* capture supported commands issued through Yarn (such as Hive, HDFS, etc). The submitApplication command is not captured.
Audit exceptions	✓	✓	Ranger only catches “access denied” exceptions. Standard S-TAP can capture other types of exceptions as well.
Redaction of returned data (Hive only)	✓	×	
Blocking of Hive	✓	✓	
Blocking of HDFS and HBASE	×	✓	
Blocking of Kafka	×	×	

Architecture and flow

In this section, we’ll go into more detail of how the integration works. We’ll start with monitoring/auditing only, as that is relatively simple and then layer in the blocking aspect.

Monitor and audit

Figure 1 below shows the basic architecture of the Guardium components alongside Ranger.

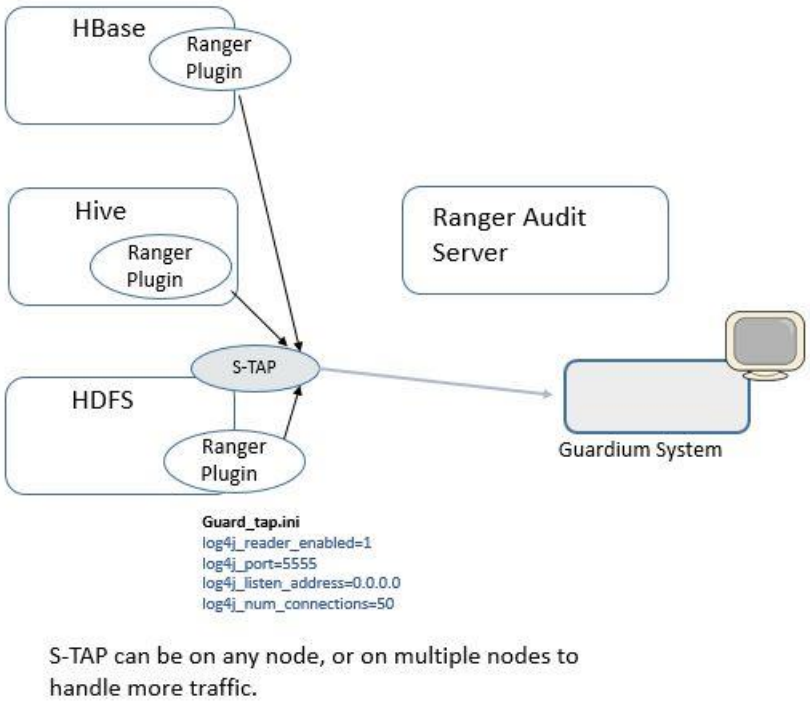


Figure 1. Ranger plugin uses log4j as an alternate logging source, which forwards audit data to the Guardium S-TAP.

The important difference between this architecture and what you may be used to with other Guardium deployments is that the S-TAP is *not* collecting audit data directly from the Hadoop component; rather, it is the Ranger plugins that are writing the audit messages to log4j, which forwards them to S-TAP, which then sends the messages to the Guardium collector for logging, alerting, reporting, and analytics.

The configuration is quite flexible in that you can install S-TAPs on more nodes. You can configure Ranger to send all component traffic to one S-TAP or you could specify, for example, that all HBase traffic goes to one S-TAP and Hive and HDFS goes to another.

In our testing in the lab, we configured one S-TAP on the name node for HDFS, Hive and Kafka traffic and one on the HBASE Master for all HBase traffic.

Blocking (Ranger Dynamic Policy integration)

Now let's layer in the blocking architecture and flow. Blocking is implemented by extending Ranger access control policies to honor blocking policy rules that are specified on the Guardium appliance. The actual implementation of blocking is performed as an access denial from Ranger.

For blocking, you need an additional component we call the *Guardium plug-in for Ranger*. This plug-in is called **Guardium_evaluator.jar** and will reside alongside the Ranger plugin on the Hadoop component nodes. Note that you will need this on the data/slave nodes as well if you want to block HBase.

S-TAPs required: You do not need any additional S-TAPs than what is already required for monitoring/auditing. It makes sense to use the same collector/S-TAP combinations for blocking as you do for auditing.

Figure 1 below shows the overall flow. A detailed description of the steps is also included below the figure.

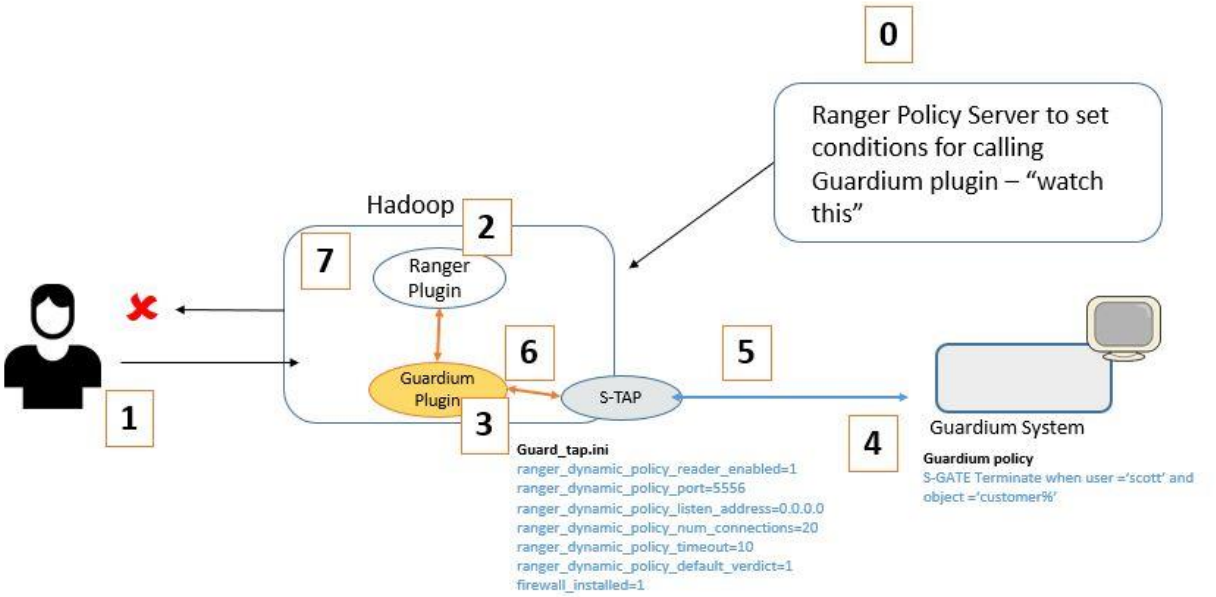


Figure 2. Blocking flow with Guardium and Hortonworks Ranger

Prerequisite (Step 0): Administrator sets up filtering conditions on a Ranger policy based on resource, user or group or other conditions allowed by Ranger. For simplicity, we call this the "watch" criteria. For example, the Ranger policy might specify Scott's activity against certain resources, because he's a privileged user. The policy also includes a condition to call the Guardium evaluator plugin. For more information about creating Ranger policies, see the Hortonworks documentation and Ranger tutorials. We also include more information in the detailed deployment steps.

The administrator sets up S-TAP to enable integration with dynamic policies and firewall. The S-TAP does not have to be directly collocated the Ranger or Guardium plugins

On the Guardium appliance, a policy is installed that includes rule action of **S-GATE Terminate** for inappropriate access to Hadoop. This rule could include additional criteria such as client IP address or other runtime information.

Here is the detailed flow:

1. User tries to access a resource that meets the "watch" criteria.
2. Ranger plugin sends information about this access to the Guardium plugin.
3. Guardium plugin sends message to S-TAP.
4. S-TAP sends request to appliance about this access.

5. If Guardium blocking policy rule conditions are met, the Guardium appliance sends “block” response to S-TAP
6. S-TAP sends “block” to Guardium plugin
7. Guardium plugin tells Ranger to **not match** the original watching rule. This means that *if there is no other Ranger policy that allows access to the resource*, then access will not be allowed to the resource.

Planning the integration

Make sure you have completed the following tasks before configuring the integration.

Topology of S-TAPs and collectors

Determine your topology, including how many collectors you need, which nodes need S-TAP, and which components each S-TAP instance will monitor. Some customers prefer to have one S-TAP for each component. At a minimum, we recommend one S-TAP for HBase and one for everything else, as shown in the figure below.

Note that S-TAP is not required to sit on the same node as any particular component. In other words, you could conceivably set up a separate Linux box that includes S-TAP. As a matter of fact, to support Hadoop HA, you may want to consider this option. See Standby deployment options, below, for more information on this topic.

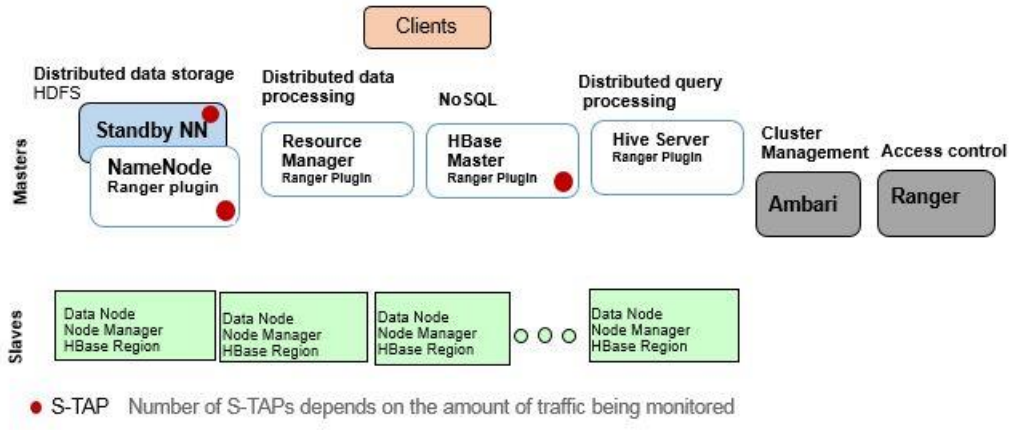


Figure 3. For Guardium auditing, a recommended initial deployment for S-TAPs

Table 2 below shows this simple mapping.

While you’re here, you might as well record the number of connections that you will need to configure for the S-TAP. A Rule of thumb is as follows:

- For HBase: 1 + number of region servers
- For everything else: 1+ 1 for each component to be monitored

Table 2. Record host names of S-TAPs and collector

Component	Node IP/Host	S-TAP node/ip	#connections	Guardium Collector Host/IP
HDFS	My.hdfs.host	My.hdfs.host	5	My.guardium.host
Hive	My.hive.host	My.hdfs.host	5	My.guardium.host
Kafka	MyKafka.host	Myhdfs.host	5	My.guardium.host
Storm	MyStorm.host	Myhdfs.host	5	My.guardum.host
HBase	MyHBase.host	My.HBase.host	51	My.guardium.host

For blocking: You also need to ensure you have access to all the HBase region servers since later on you will be copying the Guardium plugin jar file to each of these region servers.

For high availability, make sure you also record the failover node IP/host names. The next section goes into more detail on handling the failover configuration scenario.

Standby deployment options

Hadoop uses secondary nodes for high availability to handle data requests should the primary node fail. There are several options for S-TAP deployment so that you can continue to collect audit data in a failover scenario.

Model 1: Install the S-TAP and set it up on a system that is *not* part of the Hadoop cluster.

This is a very simple configuration in that, when the components fails over, the new node will automatically use the S-TAP as a remote logger. No changes need to be made to any configurations or S-TAP here.

Model 2: Install the STAP on the nodes in the cluster (not recommended)

In this model, you can install S-TAP on the primary and standby for each component.

While setting with the CLI command use "localhost" in the S-TAP host field. This means you need S-TAPs installed on every node in the cluster and every region server for HBASE.

Model 3: Hybrid approach (recommended)

In a hybrid model, you can install S-TAP with "localhost" option for HDFS and Hive, and use a separate system, such as an edge node, for HBase so that you don’t have to install S-TAPs on all nodes and region servers.

Guardium load balancing

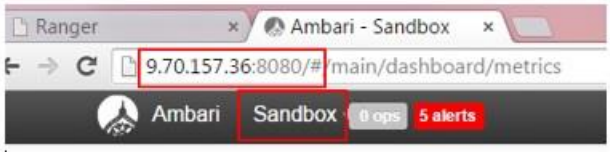
Guardium S-TAP and enterprise load balancing options are supported when Ranger integration is enabled.

Ambari and Ranger information

A significant portion of setup is done through Ambari, the Hadoop administrative interface. To complete configuration, you will need the following information:

- Ambari
    - A user ID and password who has privileges to update and save the log4j configuration, such as a Service Administrator account. For simplicity, in this document, we'll call this the admin account and password.
    - Port and IP or hostname
    - Cluster name.

    The screenshot below shows the port and IP highlighted at the top and the cluster name (Sandbox) highlighted on the bottom.



- Ranger (only needed to configure blocking)
    - Again, the Service Administrator account who can update and save the Ranger configuration.
    - Port and IP or hostname
- Host in the Hadoop cluster.
    - A user ID and password who can copy the Guardium_evaluator.jar plugin to the correct path in the Hadoop cluster. (Required for blocking only.)

Open the required ports

Ensure that the following ports are opened (assuming use of default ports):

- For monitoring, open port 5555 between the node(s) that S-TAP is on and the Ranger server.
- For blocking, open port 5556 to allow communication between S-TAP and all nodes in the cluster that have the Guardium plugin.

Configure the solution for monitoring

This section describes how to configure the solution for monitoring. In summary:

- Step 1. Configure Ranger plugins using Ambari
- Step 2. Configure Guardium and Ranger
- Step 3. Install Guardium and Ranger policies

Step 1. Configure Ranger plugins using Ambari

These are the steps to enable Ranger plug-ins for the Hadoop components you want to monitor. Refer to Hortonworks documentation for more details as needed or if you need to enable auditing on a non-Ambari cluster.

1. In Ambari, log in as the administrator. Go to **Ranger > Configs > Ranger Plugin** and enable the Ranger plugins for HDFS, Hive, Kafka and HBase. The screenshot below shows the screen in Ambari where you would do this.

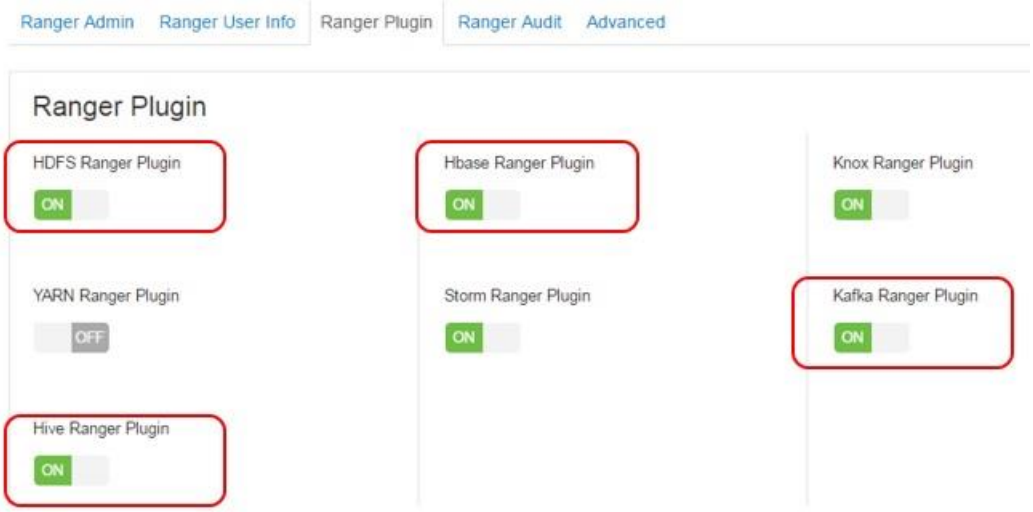


Figure 4. Enabling Ranger audit in Ambari

1. Create repositories for all the components to be audited.
2. Restart Ranger and the components.
3. Ensure Ranger auditing is turned on for each component's Ranger policies. By default, they are enabled, but you can verify in the Ranger console as shown below.

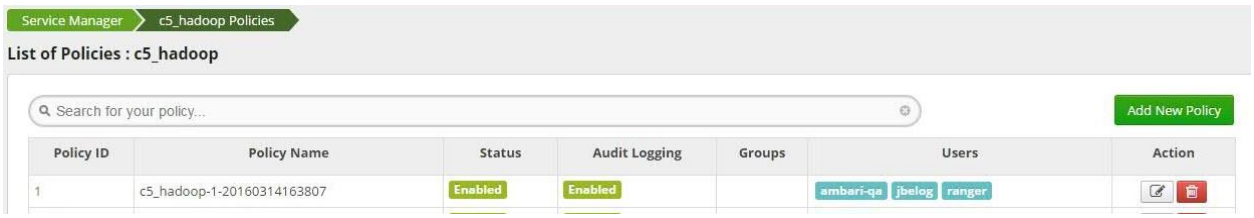
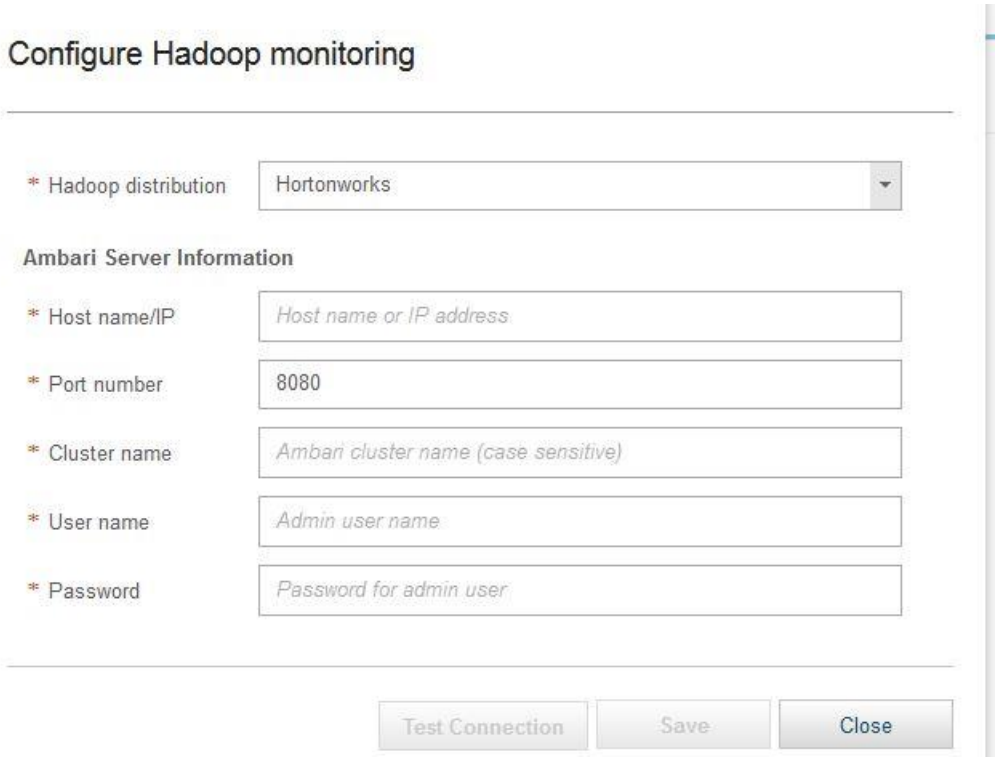


Figure 5. Validate that auditing is enabled for the Ranger policy

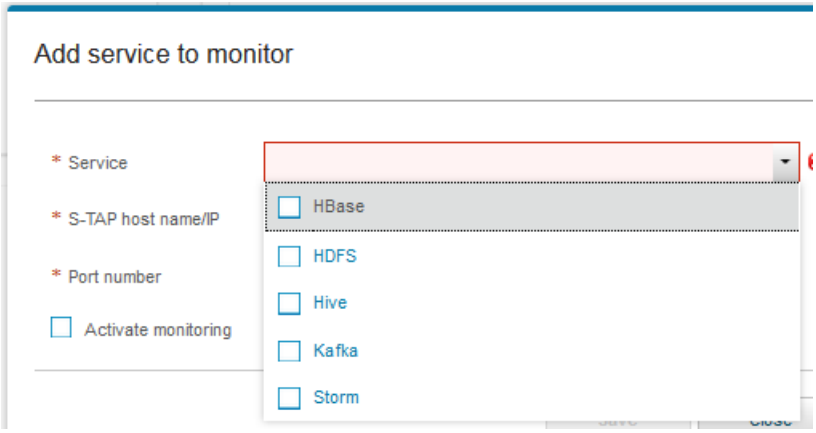
Step 2. Configure Guardium and Ranger

In this procedure you will be using the user interface (new in 10.1.2) to configure the configuration and communication between the Guardium appliance and Ranger as well as indicating which Hadoop components to monitor.

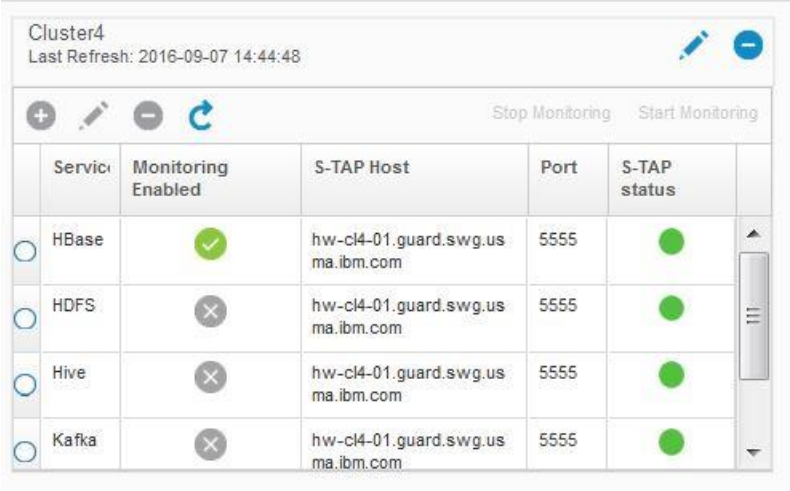
1. Ensure that the S-TAP is communicating with the collector. Check the S-TAP status monitor and ensure that the connection is green.
2. From Guardium, go to **Setup > Tools and Views > Hadoop Monitoring**
3. Click on the plus icon in the Add Cluster information and select Hortonworks from the Hadoop distribution pulldown. Your screen will look like this.



4. Add the required Ambari server information. Note that a valid Ambari admin user (or service administrator account) and password is required. Save the configuration.
5. The screen will then change so you can add which services you want to monitor and which S-TAP (or S-TAPs) will be associated with each service. To activate the components for Guardium monitoring, check the **Activate monitoring** checkbox, or you can do that later.



When you check **Activate monitoring**, Guardium will communicate with the associated Ranger plugins to enable traffic to be sent to the associated S-TAPs. In addition, there will be a green check mark on the services page to indicate that monitoring is enabled. The screenshot below, for example, shows that monitoring is enabled for HBase but not for the other services.



6. Ask the Ambari administrator to restart the activated Hadoop services in Ambari**.**
   No data can begin flowing from the Hadoop cluster to the collector until the services are restarted.

Step 3. Install Guardium and Ranger policies

Guardium relies on activated Ranger policies for data collection on the Hadoop component, which is then forwarded through the Guardium S-TAP.

The Guardium policy rules on the Guardium collector then operate on that traffic much the same as with other databases. This document assumes you have a working knowledge of Guardium policies. You may already be running with the default Guardium policy, which should enable you to see if traffic is flowing to the collector.

- See Sample Ranger policy, below, for more information on Ranger policies. See also the Hortonworks documentation.
- For a sample Guardium policy, see Sample Guardium policy, below.

Sample Ranger policy

For *Ranger*, ensure that your enabled Ranger policies do have auditing enabled. The default policy after installation and configuration of Ranger will by default audit access to all resources (that is Resource path is /*). Make sure Hive, HBase and Kafka have auditing enabled appropriately as well, assuming that you want to audit that traffic.

Here is an example of the HDFS/Hadoop Ranger policy.

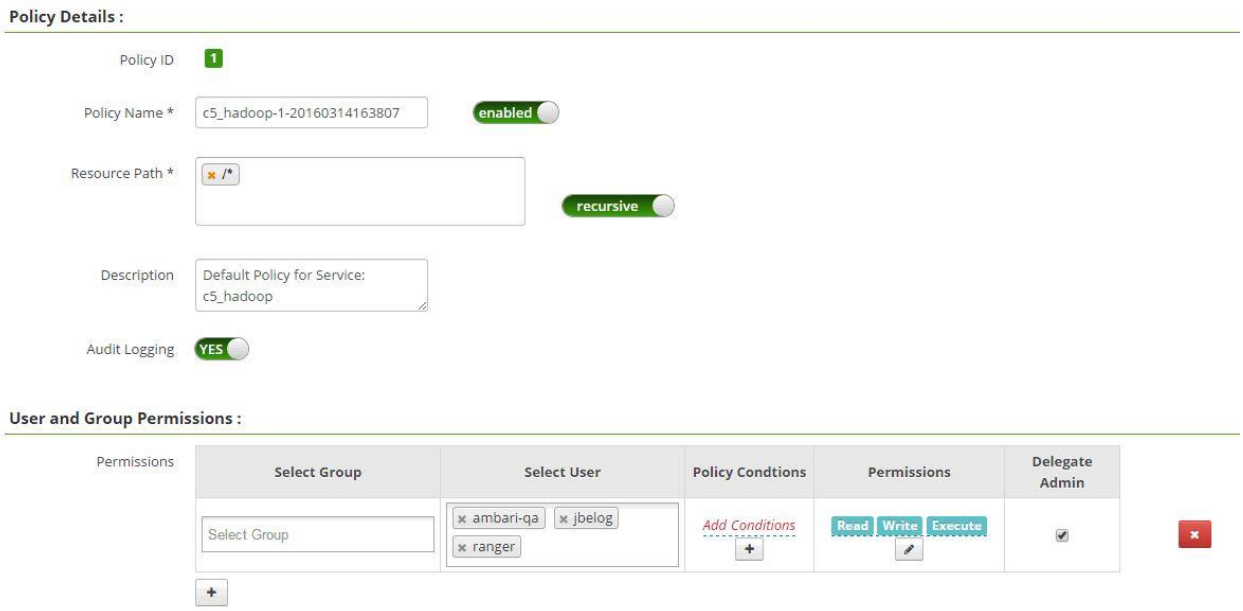


Figure 6. HDFS Ranger policy

Sample Guardium policy

For the policy used for our examples, we simply cloned the built-in Hadoop policy, removed a rule we didn't need and added additional rules we needed.

The sample policy shown in Figure 7 includes both auditing and blocking rules for the purpose of illustration and so that we could validate the rules working together in our test environment. However, it is strongly recommended that you implement monitoring and get it working as desired before attempting blocking. Blocking should be implemented as a separate project and we describe the blocking rules in more detail in Step 5: Configure Ranger and Guardium policies in the Blocking section of this document.

An important thing to note is that for *auditing*, policy rules will be the same as if using S-TAP inspection engines. (There are minor differences in how Hadoop commands are sent to Guardium and we'll cover some of that here.)

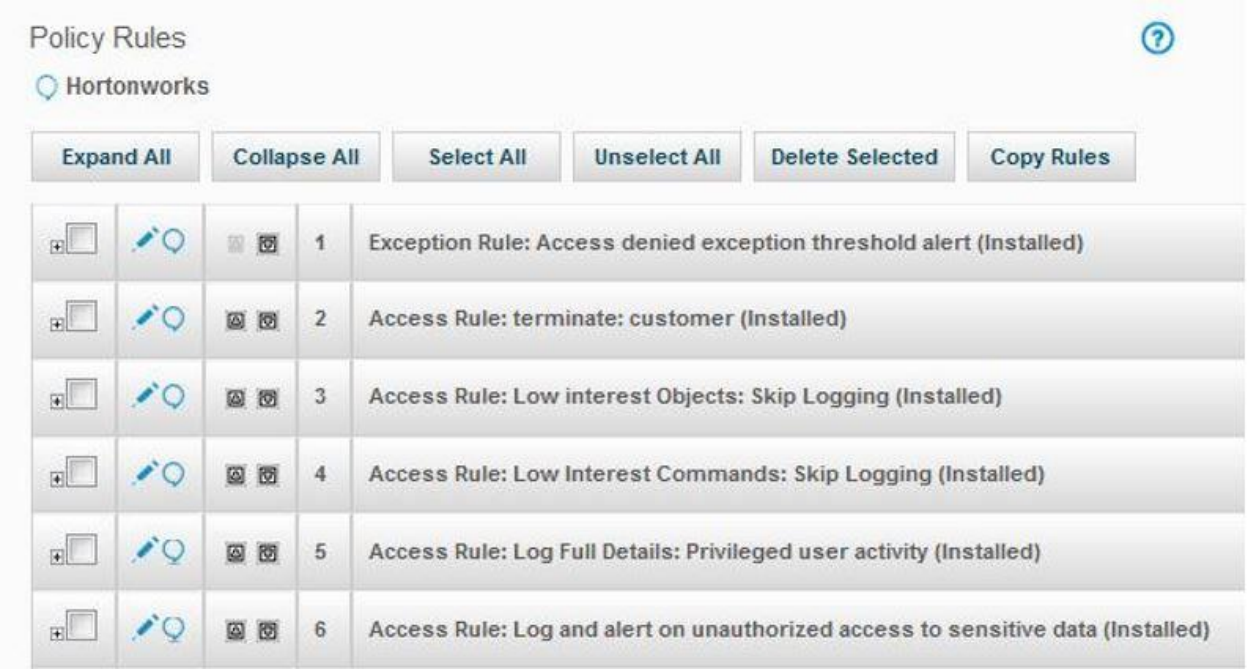


Figure 7. A sample Guardium policy for Hadoop

From a high level perspective, this policy is doing the following (rules are evaluated in order)

- Alerting on excessive access denied exceptions
- Blocking access to customer data from a privileged user. This rule requires more extensive explanation and is covered in the Blocking section of this document.
- Skipping some logging for noise commands that are of no relevance for auditing. (The skip logging rules are included in the Hadoop default policy included in Guardium and was discussed in more detail in the *Hadoop Deployment Guide* as well.)
- Logging all activity in detail for privileged users
- Logging full detail and alerting any access to sensitive data from users not in the production user list.
- Everything else will be logged using the default logging (constructs only).

Access denied threshold (Exception rule)

Here are the relevant fields from the exception rule. Note that access denied exceptions are captured as an SQL ERROR type of exception. This is one case that does differ from inspection engines in that Ranger only sends “access denied” exceptions, whereas inspection engines would pick up other types of exceptions such as disk failures.

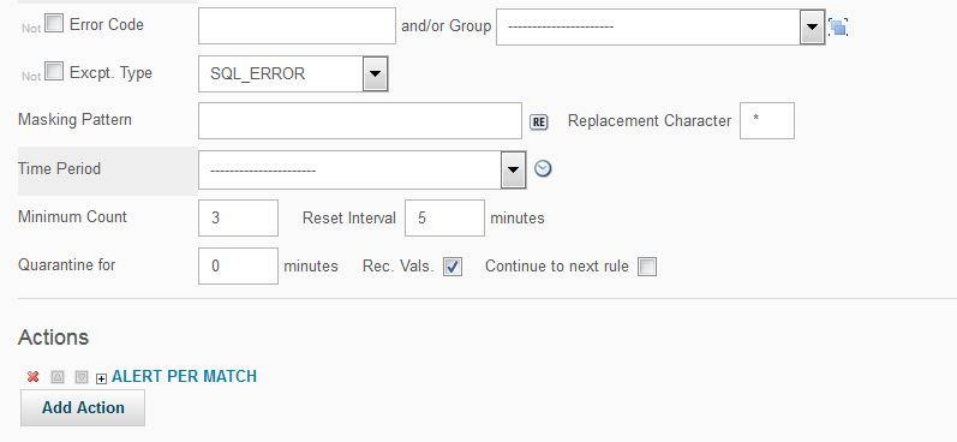


Figure 8. Exceptions policy rule

Here is what the violation looks like in the Policy Violations report.

	189263000000000055	2016-06-14 22:17:44		Access denied exception threshold alert	9.70.156.60	hw-cl5-01.guard.swg.usma.ibm.com	SVORUGA	EXECUTE path=/user/svoruga /plans/secret /secret1.txt._COPYING_	INFO

Privileged user activity (Access rule)

In this case, we want to log the full details of any activity our privileged users do on the system. Figure 9 shows the relevant fields in the policy and our privileged user group members.

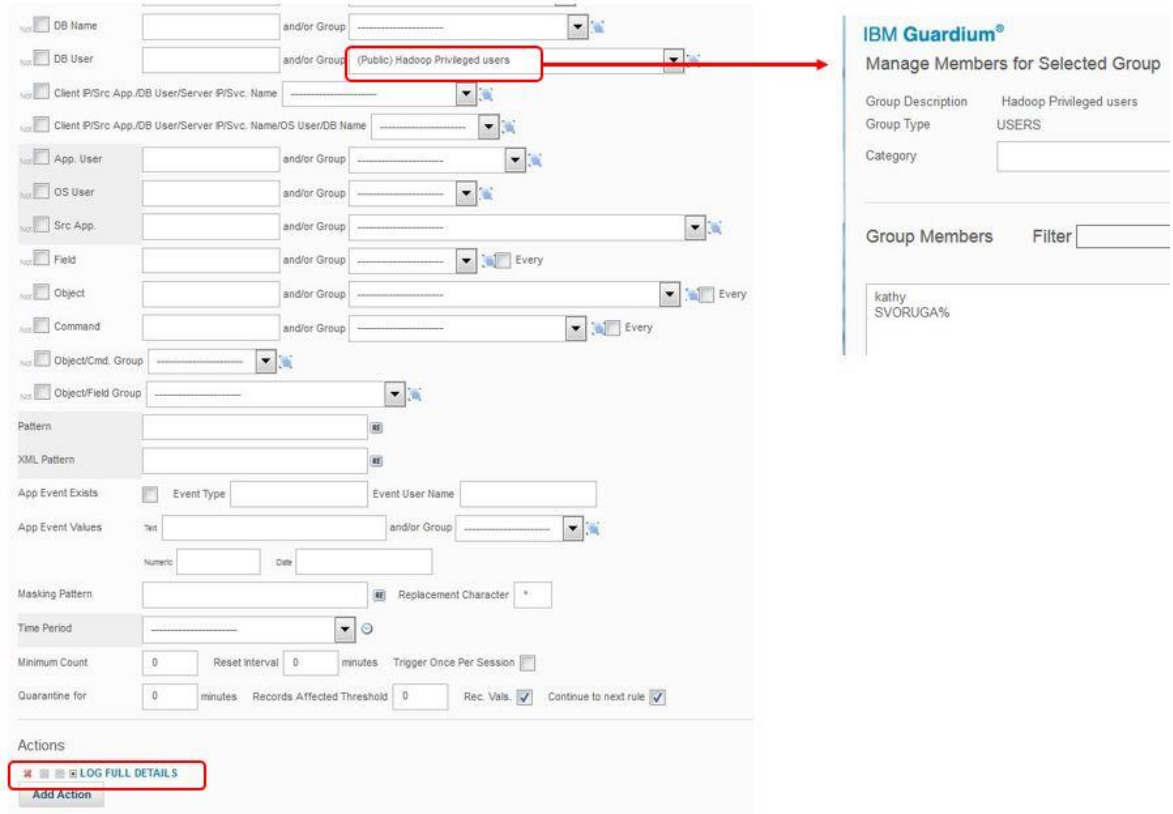


Figure 9. Privileged user activity policy rule

Figure 11 is an example of report output from the default Hadoop –privilege user report.

Prerequisite: This report uses the built in Hadoop Server types group. Make sure you installed the DPS that came with the 10.1.2 GPU, or modify the group members with the services you are monitoring. An example is shown in Figure 11.

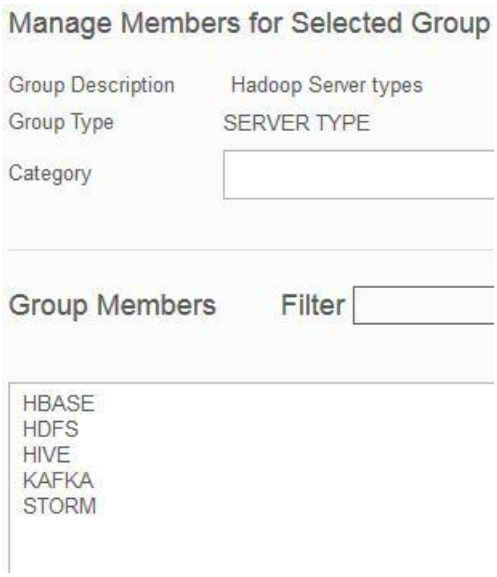


Figure 10. Add Kafka and HDFS to the Hadoop Server types group

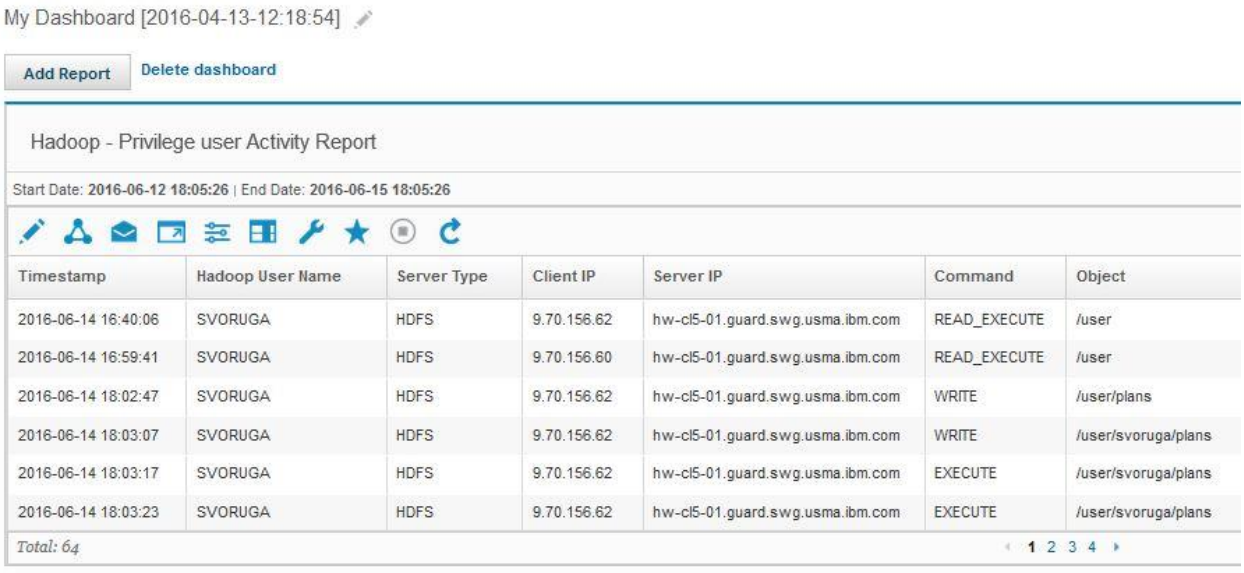


Figure 11. Sample output for privileged user activity

Log and alert on unauthorized access to sensitive data (Access rule)

In this rule, we are using a black list by specifying a NOT condition so that any access other than our production users (such as IDs used by vetted applications) would have access to sensitive data.

This rule both logs the full details of the access and sends an alert.

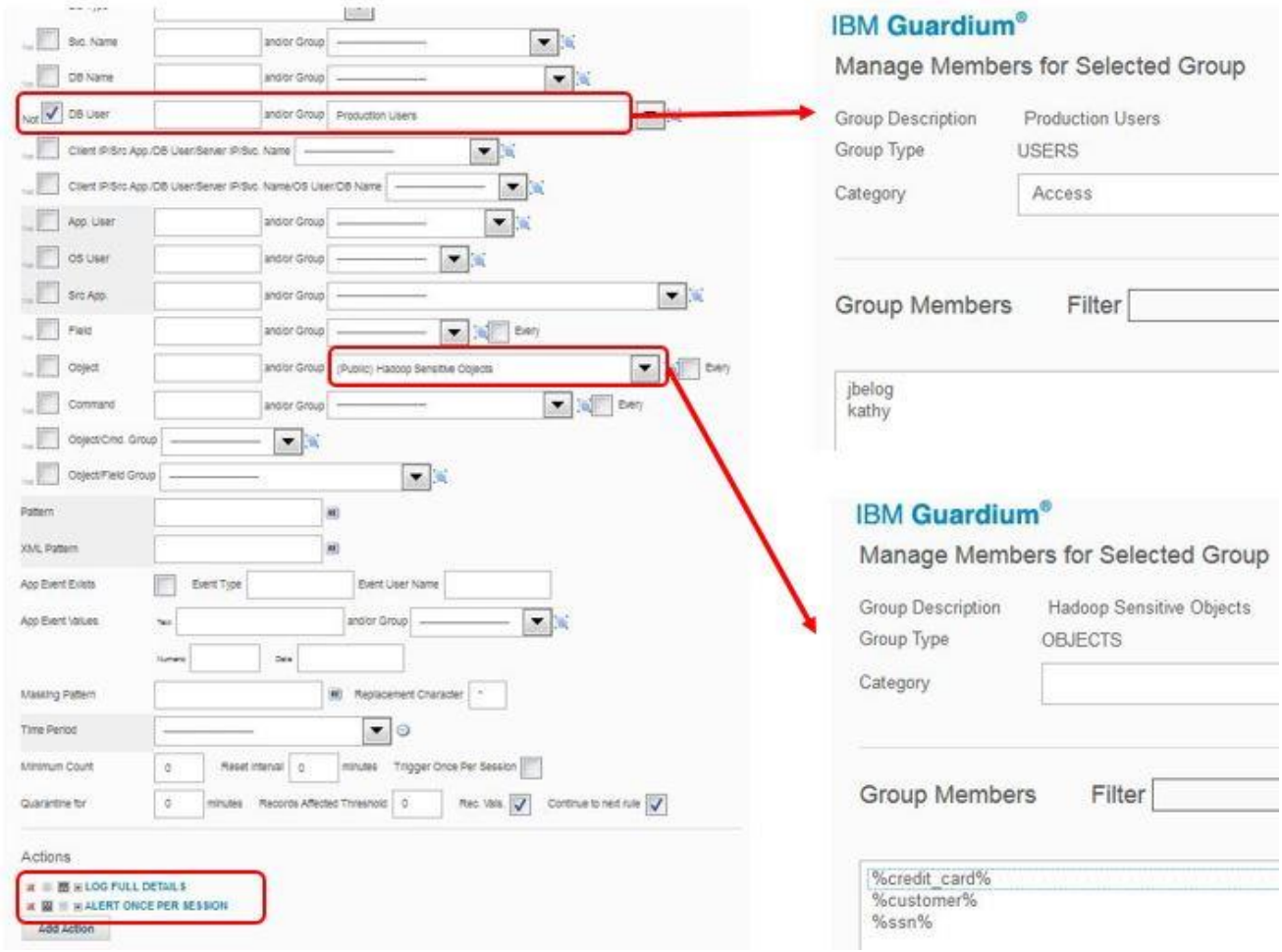


Figure 12.Policy rule to alert on unauthorized access to sensitive data

Here is what the violation looks like in the Policy Violations report when SVORUGA, who is not a member of our production user group, accesses customer data.

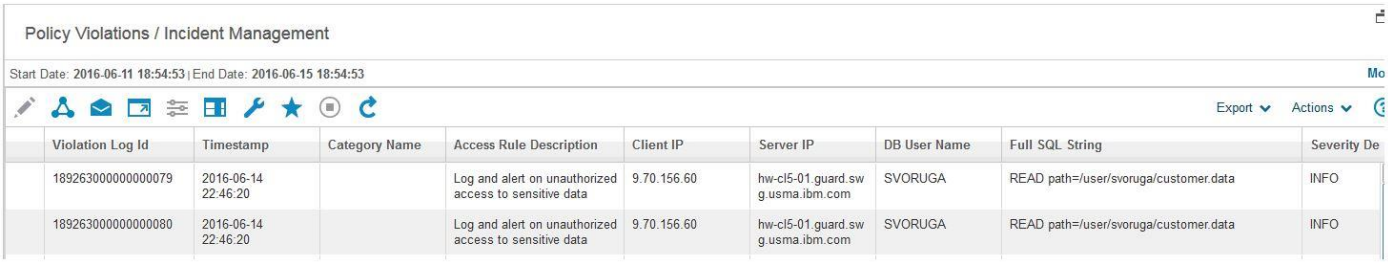


Figure 13. Violation

Standard audit (Access rule)

The last rule in the policy has no conditions and will fire for any access that does not fall into any of the above categories. This rule log constructs only (no full sql). Ideally, you would put some kind of limit on this such as by data resource, but if you do need to do some level of auditing for all access, you would use an empty rule with ALLOW as the action.

Configure the solution for blocking

Important: Blocking integration is complex and must be configured carefully to avoid unintended consequences such as impacting system performance. It requires coordination of policy information on both the Ranger and Guardium side and policy rules have to be in the right order on Ranger.

HDFS blocking with Guardium requires that files that must be blocked have ALL permissions removed.

Here are the steps:

- Step 1. Enable the components for blocking
- Step 2. Copy the Guardium plug-in for Ranger to appropriate directory
- Step 3. Restart the Hadoop and Ranger components
- Step 4. Configure the guard_tap.ini parameters
- Step 5: Configure Ranger and Guardium policies

Step 1. Enable the components for blocking

In this step, you will be running a Python script (ranger_dynpolicy_config.py) to tell the Ranger server that Guardium is using dynamic policy support for these components. This script requires network access to the Ranger server.

The parameters for the command are:

Parameter	**Description**
-a	Ranger host

-b	Ranger port
-u	Admin ID for Ranger
-p	Password for Ranger Admin
-l	Dynamic policy port. Must match the port specified in the guard_tap.ini file.
-s	Component to enable for blocking. Valid values are: • hdfs • hbase • hive Only one component can be specified for each invocation of the script.
-x	Enable/Disable activation.

Here is an example that configures HDFS for blocking. Log in as root or guardium and run this on the node(s) where S-TAP is installed.

```
/usr/local/guardium/guard_stap/ranger_dynpolicy_config.py -a hw-cl5-
01.guard.swg.usma.ibm.com -b 6080 -u admin -p admin -l 5556 -s hdfs -x enable
```

Step 2. Copy the Guardium plug-in for Ranger to appropriate directory

Obtain the file **guardium_evaluator.jar** from the guard_stap directory of any machine that has a 10.1 UNIX S-TAP installed on it.

Copy this file to the following locations:

- The Ranger webapp directory as follows:
  /usr/hdp/current/ranger-admin/ews/webapp/WEB-INF/lib/

- The Ranger plugin directory for each applicable service for which Guardium blocking is required, as follows:

  - HDFS: /usr/hdp/<current version>/hadoop/lib/ranger-hdfs-plugin-impl/
  - Hive: /usr/hdp//<current version>/hive/lib/ranger-hive-plugin-impl
  - HBase: /usr/hdp/<current version>/hbase/lib/ranger-hbase-plugin-impl
    Important: For Hbase, the Guardium jar must be installed on both master *and* region servers.

Step 3. Restart the Hadoop and Ranger components

After you do the above steps, restart both the Hadoop components (HDFS, Hive, HBase…) and Ranger.

Step 4. Configure the guard_tap.ini parameters

For blocking, you must edit some parameters in guard_tap.ini. You must edit the file directly; there is no UI or GIM settings for this.

Important: Restart the S-TAP after modifying the parameters.

Below is an example in which the parameters have already been set to enable blocking.

ranger_dynamic_policy_reader_enabled=1

ranger_dynamic_policy_port=5556

ranger_dynamic_policy_listen_address=0.0.0.0

ranger_dynamic_policy_num_connections=20

ranger_dynamic_policy_timeout=10

ranger_dynamic_policy_default_verdict=1

The description of these parameters is as follows:

Parameter	**Description**
firewall_enabled	Enables and disables blocking. 0-=no (the default) 1=yes **Note**: You can still use 'regular' blocking for non-Ranger traffic.
ranger_dynamic_policy_reader_enabled	Enable Hadoop blocking by enabling the use of Ranger dynamic policies. 0=no (the default) 1=yes
ranger_dynamic_policy_port	The port that Guardium S-TAP will use to communicate with the Guardium plug-in for Ranger. The default is 5556. This must be different than the ranger_log4j port value.

ranger_dynamic_policy_listen_address	This is the address that the Ranger plugins will try to connect to. 0.0.0.0 means any address of this machine. Localhost means only listen on the loopback network device on the machine. The default and recommended value is 0.0.0.0, which enables S-TAP to receive traffic from any host. Use localhost if configuring the system for high availability as described above in Standby deployment options.
ranger_dynamic_policy_num_connections	The maximum number of connections to support from the Guardium plugin for Ranger. The default is 20.
ranger_dynamic_policy_timeout	The number of seconds to wait for a verdict from the Guardium appliance before sending the default verdict result. The default is 10.
ranger_dynamic_policy_default_verdict	What to do if a request for a verdict can't be sent to the Guardium appliance or if the timeout is reached before a verdict is received. 1=default, block (deny access) 0= allow.

Step 5: Configure Ranger and Guardium policies

This configuration requires a coordinated effort between setting up the filtering conditions on a Ranger policy for a specific component, and then configuring the conditions for the blocking on Guardium. Any activity that passes the initial Ranger policy condition will be checked against the Guardium policy to see if it should be blocked. When the Guardium plugin for Ranger receives the verdict from the appliance, it will pass that onto Ranger.

Important: The policies that you have defined in Ranger need to be in the right order.

An example for HDFS

We'll use a relatively simple example to illustrate how the interaction between Guardium and Ranger works for blocking.

Prerequisites:

Remember to *remove all permissions* for files to be blocked. To remove permissions, use the command - chmod 000.

In this example, we want to be sure that our privileged user, SVORUGA, cannot access customer data, so all resources with Customer in the name must have no permissions.

Example:
[hdfs@sandbox ~]$ hadoop fs -chmod 000 /user/svoruga/customer.data
[hdfs@sandbox ~]$ hadoop fs -ls /user/svoruga

Found 1 items
---------- 3 hdfs hdfs 49 2017-08-31 21:15 /user/svoruga/customer.data

On the Ranger side, we set up policy to watch all activity in the svoruga directory by svoruga. Edit the guard-plugin: Block to enable it.

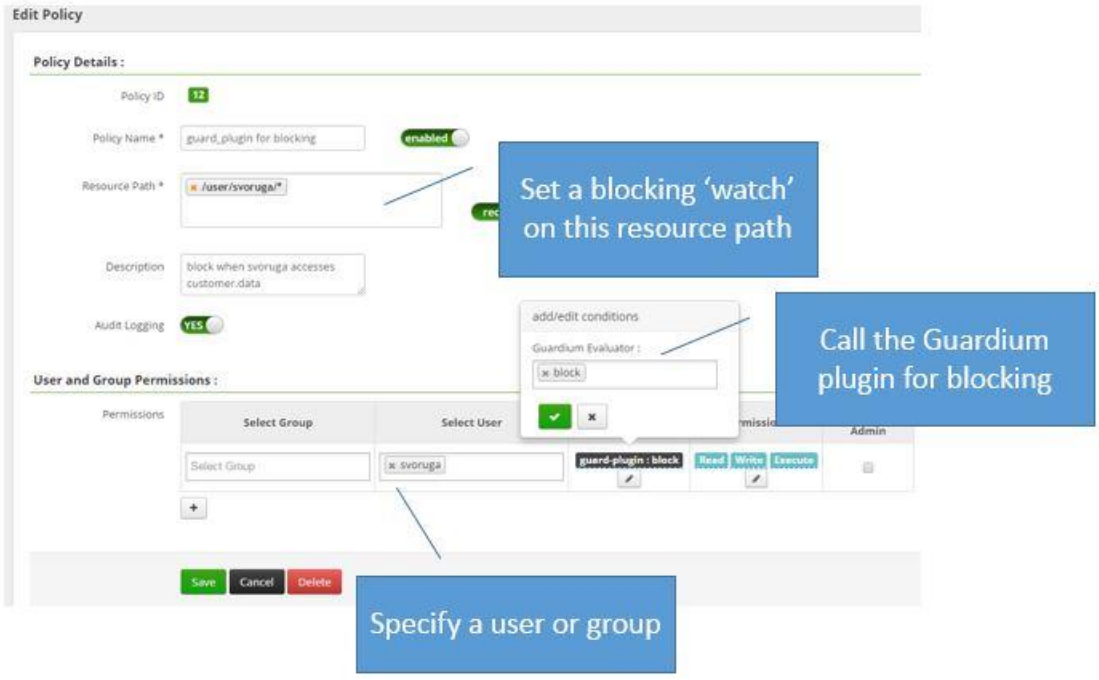


Figure 14. Ranger policy extended with Guardium plugin for Ranger

Now, on the Guardium side, we just need to specify any additional conditions to actually block on. In this example, we're narrowing it down to customer objects. Note that unlike other Guardium blocking rules, do not specify an attach rule. You are already doing that with the Ranger dynamic policy integration.

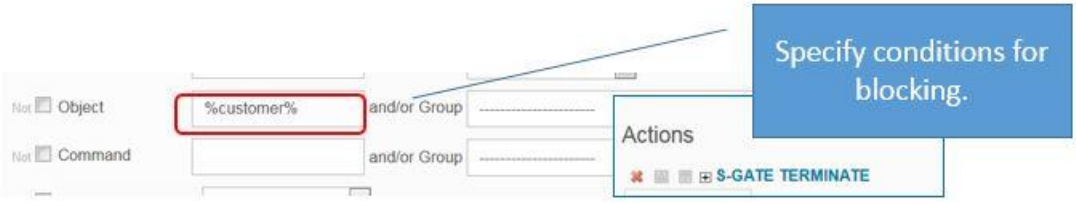


Figure 15.Guardium policy specifies conditions to block

So, now assume that svoruga tries to cat the customer.data file in that svoruga directory. She will get a permission denied message as shown here.

```
$0-svoruga@hw-cl4-01:~> hadoop fs -cat /user/svoruga/customer.data
cat: Permission denied: user=svoruga, access=READ, inode="/user/svoruga/customer
.data":svoruga:hdfs:----------
```

This is what it looks like in Guardium policy violations report:

And you'll see the access denied in Ranger as well.

Hive and HBase examples

Compared to HDFS, Hive and HBase are simpler in that you don't need to worry about removing permissions on the files.

For example, assume we want to block access from guest users and our privileged user svoruga to customer data in the demo database. We can use Ranger to set up the watch on the Demo database for those users and then use the Guardium policy to specify blocking for any customer objects in that database.

Here's the Ranger policy calling the Guardium plug in when guest users or svoruga accesses the Demo database.

Figure 16.Ranger policy for Hive calling the Guardium plugin

And we use the same rule as we did for HDFS to do the blocking on customer objects. See Figure 15, above.

Now, when svoruga or guest users do the following from beeline:

```
Use demo;
```

```
Select * from customer;
```

They will get access denied.

For HBase, here is the Ranger policy.

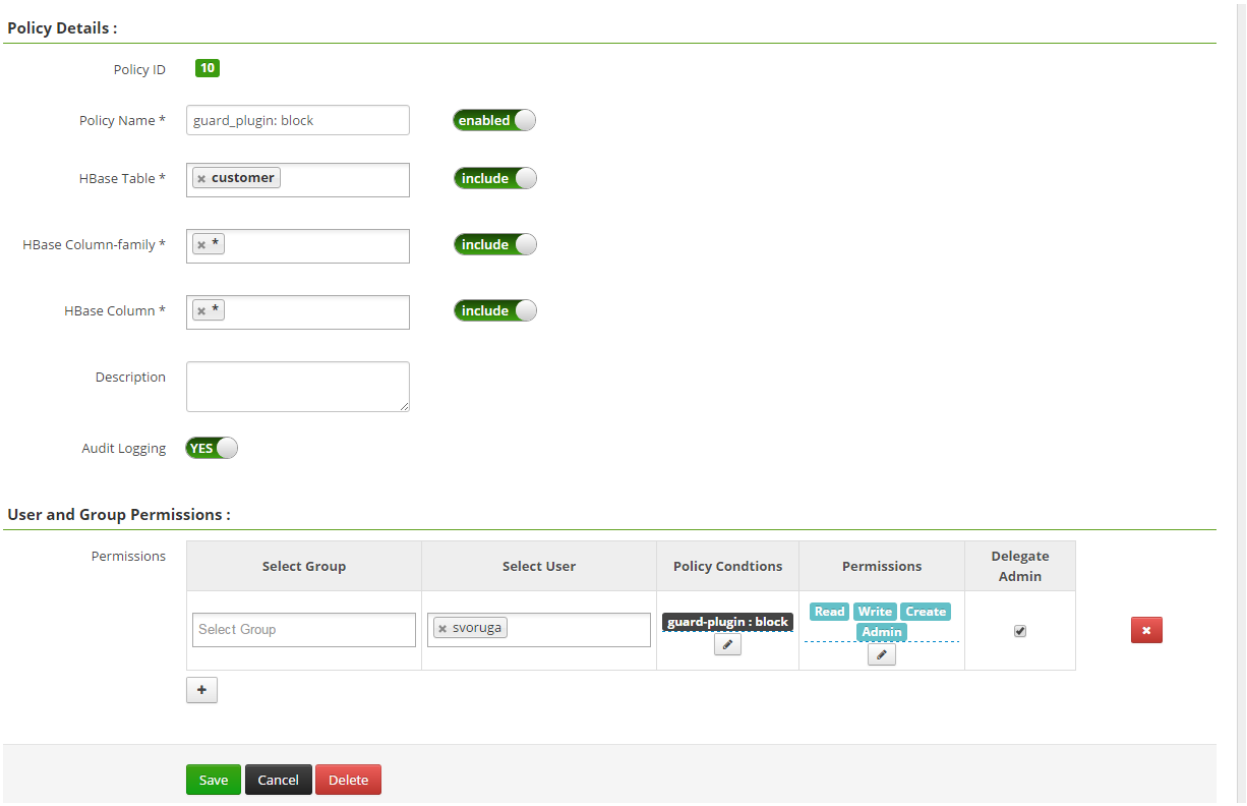


Figure 17.Ranger policy watching svoruga's access on customer table in HBase

Now if svoruga logs in and runs

```
scan 'customer'
```

She'll get an access denied exception.

Here is what it looks lke in the policy violations report.

	Violation Log Id	Timestamp	Category Name	Access Rule Description	Client IP	Server IP	DB User Name	Full SQL String
	1892630000000000005	2016-04-12 20:42:01		terminate:customer	9.70.156.62	hw-cl5-03.guard.swg.usma.ibm.com	SVORUGA	getTableDescriptors table=customer
	1892630000000000004	2016-04-12 20:41:37		terminate:customer	9.70.156.62	hw-cl5-04.guard.swg.usma.ibm.com	SVORUGA	scannerOpen column-family,table=cf1,customer

Information about commands

This section provides some sample commands for the various components and how they appear in Guardium when using the Ranger integration.

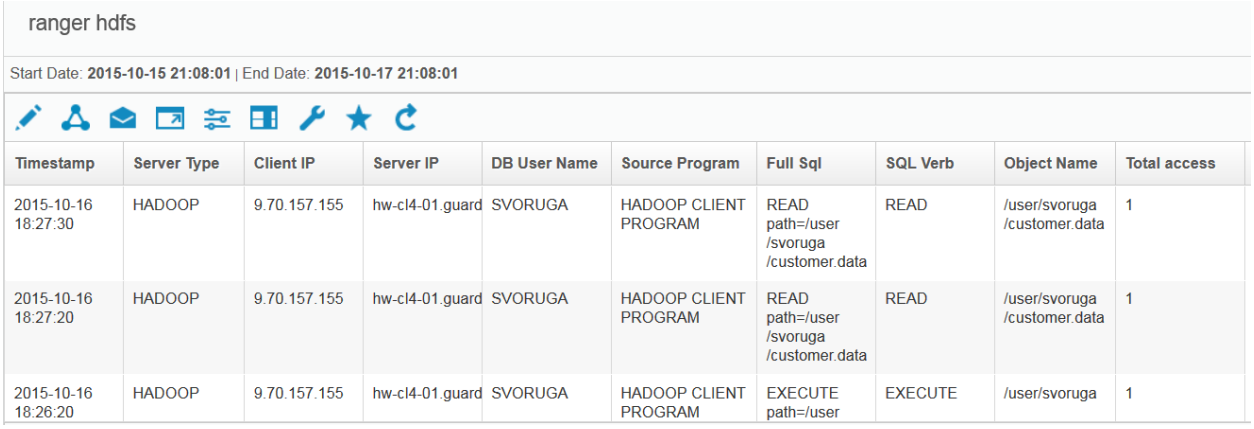
HDFS commands and sample activity report

Ranger is picking up low level file commands. This table shows a sample of some commands and how they would appear in Guardium. The full command is shown as if you were doing a FullSQL report. Otherwise, only the Command/Verb (READ, WRITE, and READ_EXECUTE) is shown in the report.

Table 3. How selected HDFS commands are logged in Guardium (from Ranger integration)

Hadoop Command	**How it appears in Guardium Activity (Full SQL)**
-ls /tmp/jhung	READ_EXECUTE path=/tmp/jhung
-mkdir /tmp/jhung	WRITE path=/tmp/jhung
-cat /tmp/jhung/jhung_hadoop	READ path=/tmp/jhung/jhung_hadoop
-tail /tmp/jhung/jhung_hadoop	READ path=/tmp/jhung/jhung_hadoop
-mv /tmp/jhung/jhung_hadoop /tmp/jhung/jhung_moved	WRITE path=/tmp/jhung/jhung_moved
-get /tmp/jhung/jhung_moved	READ path=/tmp/jhung/jhung_moved
-cp /tmp/jhung/jhung_moved /tmp/jhung/jhung_copied	READ path=/tmp/jhung/jhung_moved WRITE path=/tmp/jhung/jhung_copied._COPYING_ WRITE path=/tmp/jhung/jhung_copied
-copyToLocal /tmp/jhung/jhung_copied /root	READ path=/tmp/jhung/jhung_copied
-copyFromLocal /root/jhung_hadoop /tmp/jhung/jhung_copiedFromLocal	WRITE path=/tmp/jhung/jhung_copiedFromLocal._COPYING_ WRITE path=/tmp/jhung/jhung_copiedFromLocal._COPYING_ WRITE path=/tmp/jhung/jhung_copiedFromLocal
-rm -r /tmp/jhung	EXECUTE path=/user/svoruga/.Trash

	WRITE path=/user/svoruga/.Trash/Current/tmp WRITE path=/user/svoruga/.Trash EXECUTE path=/user/svoruga/.Trash/Current/tmp WRITE path=/user/svoruga/.Trash/Current/tmp/jhung WRITE path=/user/svoruga/.Trash/Current/tmp WRITE path=/tmp/jhung
-put jhung_hadoop /tmp/jhung	WRITE path=/tmp/jhung/jhung_hadoop._COPYING_ WRITE path=/tmp/jhung/jhung_hadoop



Timestamp	Server Type	Client IP	Server IP	DB User Name	Source Program	Full Sql	SQL Verb	Object Name	Total access
2015-10-16 18:27:30	HADOOP	9.70.157.155	hw-cl4-01.guard	SVORUGA	HADOOP CLIENT PROGRAM	READ path=/user /svoruga /customer.data	READ	/user/svoruga /customer.data	1
2015-10-16 18:27:20	HADOOP	9.70.157.155	hw-cl4-01.guard	SVORUGA	HADOOP CLIENT PROGRAM	READ path=/user /svoruga /customer.data	READ	/user/svoruga /customer.data	1
2015-10-16 18:26:20	HADOOP	9.70.157.155	hw-cl4-01.guard	SVORUGA	HADOOP CLIENT PROGRAM	EXECUTE path=/user	EXECUTE	/user/svoruga	1

Figure 18. HDFS activity in Guardium.

Hive commands and sample activity report

Hive traffic looks very similar to SQL traffic you see from other databases.

Table 4. Hive commands in Guardium

Hive Command	**Command (Verb)**
create database retail;	CREATE DATABASE
use retail;	USE DATABASE
create table txnrecords(txnno INT, txndate STRING, amount DOUBLE);	CREATE TABLE
describe txnrecords;	DESCRIBE

insert into table txnrecords values (32, '010295', 11.5);	INSERT
select * from txnrecords;	SELECT
GRANT SELECT ON txnrecords to USER guest;	GRANT
REVOKE SELECT ON txnrecords FROM USER guest;	REVOKE
drop table txnrecords;	DROP TABLE
drop database retail;	DROP DATABASE

The Full SQL report below shows that a GUEST user ran a CREATE TABLE and then ran an INSERT statement that includes an INSERT with subselect. Because there are three objects referenced in that INSERT statement, there are three lines in the report.

Timestamp	Server Type	Client IP	Server IP	DB User Name	SQL Verb	Object Name	Full Sql
2016-04-20 21:39:49	HIVE	9.70.156.60	hw-cl5-03.guard.swg .usma.ibm.com	GUEST	INSERT	events1	INSERT OVERWRITE TABLE events1 SELECT a.bar, count(*) FROM invites a WHERE a.foo > 0 GROUP BY a.bar
2016-04-20 21:39:49	HIVE	9.70.156.60	hw-cl5-03.guard.swg .usma.ibm.com	GUEST	SELECT	count	INSERT OVERWRITE TABLE events1 SELECT a.bar, count(*) FROM invites a WHERE a.foo > 0 GROUP BY a.bar
2016-04-20 21:39:49	HIVE	9.70.156.60	hw-cl5-03.guard.swg .usma.ibm.com	GUEST	SELECT	invites	INSERT OVERWRITE TABLE events1 SELECT a.bar, count(*) FROM invites a WHERE a.foo > 0 GROUP BY a.bar
2016-04-20 21:39:41	HIVE	9.70.156.60	hw-cl5-03.guard.swg .usma.ibm.com	GUEST	CREATE TABLE	events1	CREATE TABLE events1 (foo INT, bar STRING)

Figure 19. Report example: Hive activity

HBase commands and sample activity report

When users run commands, it will likely be reported under the original requester (DB user) but also have many operations underneath running as HBase.

For example, assume user JBELOG logs in and enters the following command:

create 'jhung', 'f1', 'f2', 'f3'

Range/Guardium will capture the following activity in full SQL (if you specify a Guardium policy with Log full details actions)

Under user JBELOG:

createTable table=jhung

Under user HBASE:

open table=jhung

For this reason, consider filtering out the HBASE DB User from your reports for auditing purposes.

In the table below, we've included the logged commands (Verbs), not the full SQL. In cases where HBASE is running an 'open' to fulfill the request, we've not included that.

Table 5. HBase commands in Guardium

HBase Command	**Command (Verb)**
create 'jhung', 'f1', 'f2', 'f3'	createTable
put 'jhung', 'r1', 'f1', 'value1'	put
get 'jhung', 'r1'	get (possibly multiple gets depending on the structure and values)
alter 'jhung', {NAME => 'f4'}	getTableDescriptors addColumn
alter 'jhung', {NAME => 'f4', METHOD => 'delete'}	getTableDescriptors getTableDescriptors deleteColumn
alter 'jhung', NAME => 'f1', VERSIONS => 5	modifyColumn getTableDescriptors
scan 'jhung'	scannerOpen (multiple scanners)
count 'jhung'	scannerOpen (multiple scanners)
delete 'jhung', 'r1', 'f1'	delete
disable 'jhung'	disableTable
grant 'jhung', 'RW', 'jhung', 'f1', 'c1'	grant getTableDescriptors
revoke 'jhung', 'jhung', 'f1', 'c1'	revoke getTableDescriptors
drop 'jhung'	deleteTable

In the report example below, we've sorted the report by DB User Name column so you can get a better picture of what the user actually did. You can choose to filter out the HBase user by modifying the report query.

ranger hbase

Timestamp	Server Type	Client IP	Server IP	DB User Name	SQL Verb	Object Name	Full Sql
2016-06-30 22:04:46	HBASE	9.70.156.62	hw-cl5-03.guard.swg.usma.ibm.com	JBELOG	createTable	jhung2	createTable table=jhung2
2016-06-30 22:11:25	HBASE	9.70.156.62	hw-cl5-03.guard.swg.usma.ibm.com	JBELOG	addColumn	jhung	addColumn table=jhung
2016-06-30 22:11:25	HBASE	9.70.156.62	hw-cl5-03.guard.swg.usma.ibm.com	JBELOG	getTableDescriptors	jhung	getTableDescriptors table=jhung
2016-06-30 22:11:29	HBASE	9.70.156.62	hw-cl5-03.guard.swg.usma.ibm.com	JBELOG	getTableDescriptors	jhung	getTableDescriptors table=jhung
2016-06-30 22:12:30	HBASE	9.70.156.62	hw-cl5-03.guard.swg.usma.ibm.com	JBELOG	getTableDescriptors	jhung	getTableDescriptors table=jhung
2016-06-30 22:12:30	HBASE	9.70.156.62	hw-cl5-03.guard.swg.usma.ibm.com	JBELOG	deleteColumn	jhung	deleteColumn table=jhung
2016-06-30 22:13:39	HBASE	9.70.156.62	hw-cl5-03.guard.swg.usma.ibm.com	JBELOG	modifyColumn	jhung	modifyColumn table=jhung

Figure 20. Report example: HBase activity (sorted on DB User name)

Kafka commands and sample activity report

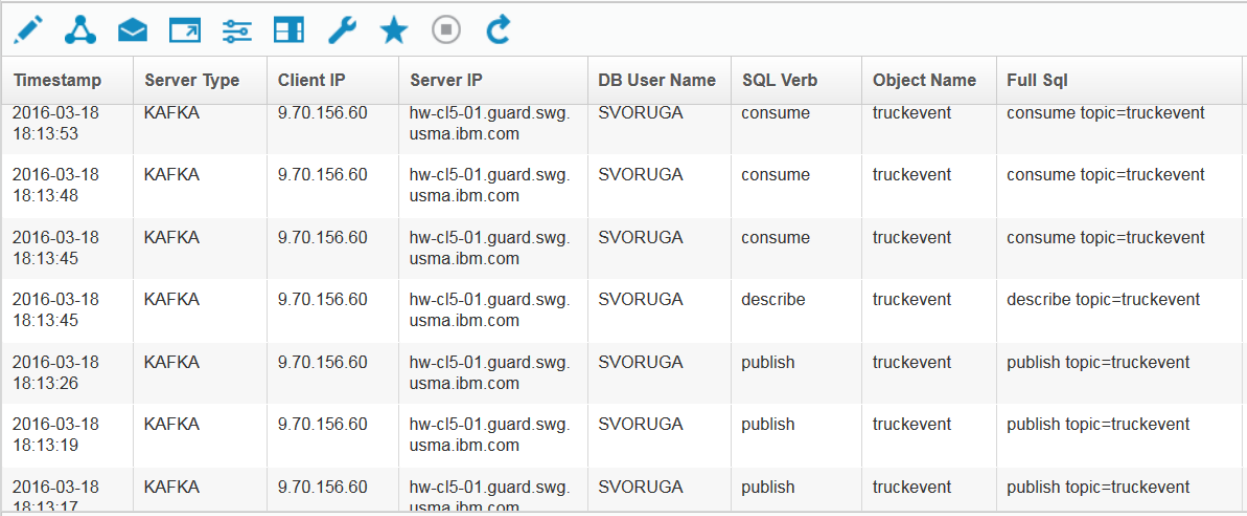
According to the Apache Kafka web site, Kafka is a distributed, partitioned, replicated commit log service that provides the services of a messaging system. From an auditing perspective you may find it helpful to learn some of the terminology such as the categorization of message feed, called *topics*. Processes can publish message to a Kafka topic and other processes can subscribe to those topics and process the message feeds.

Our testing is done based on the following Hadoop tutorial, which shows how to use Kafka to process real-time event data from trucks. Sensors report real-time events like speeding, lane-departure, unsafe tailgating, and unsafe following distances.

http://hortonworks.com/hadoop-tutorial/simulating-transporting-realtime-events-stream-apache-kafka/

The commands for Kafka auditing are very simple:

- Publish (write an event to a topic)
- Consume (read the event from a topic)
- Describe (internally occurs before a process consumes a message)



Timestamp	Server Type	Client IP	Server IP	DB User Name	SQL Verb	Object Name	Full Sql
2016-03-18 18:13:53	KAFKA	9.70.156.60	hw-cl5-01.guard.swg.usma.ibm.com	SVORUGA	consume	truckevent	consume topic=truckevent
2016-03-18 18:13:48	KAFKA	9.70.156.60	hw-cl5-01.guard.swg.usma.ibm.com	SVORUGA	consume	truckevent	consume topic=truckevent
2016-03-18 18:13:45	KAFKA	9.70.156.60	hw-cl5-01.guard.swg.usma.ibm.com	SVORUGA	consume	truckevent	consume topic=truckevent
2016-03-18 18:13:45	KAFKA	9.70.156.60	hw-cl5-01.guard.swg.usma.ibm.com	SVORUGA	describe	truckevent	describe topic=truckevent
2016-03-18 18:13:26	KAFKA	9.70.156.60	hw-cl5-01.guard.swg.usma.ibm.com	SVORUGA	publish	truckevent	publish topic=truckevent
2016-03-18 18:13:19	KAFKA	9.70.156.60	hw-cl5-01.guard.swg.usma.ibm.com	SVORUGA	publish	truckevent	publish topic=truckevent
2016-03-18 18:13:17	KAFKA	9.70.156.60	hw-cl5-01.guard.swg.usma.ibm.com	SVORUGA	publish	truckevent	publish topic=truckevent

Figure 21. Kafka activity report.

Storm activity and sample report

Storm is real-time computation system used for processing streams of data. A Storm topology consumes and processes streams of data and may repartition the streams between each stage of the computation.

The node in a topology that does some processing and then emits a stream is called a “spout”.

Here are some sample commands. The resulting activity in Guardium is shown in Figure 22.

This command submits the topology:

```
storm jar /usr/hdp/current/storm-client/contrib/storm-starter/storm-starter-topologies-
0.10.0.2.3.4.0-3485.jar storm.starter.WordCountTopology wordcount12 -c
java.security.auth.login.config=/etc/storm/conf/client_jaas.conf -c
storm.thrift.transport=backtype.storm.security.auth.kerberos.KerberosSaslTransportPlugin
```

This command is killing the specified topology:

```
storm kill wordcount15 -c java.security.auth.login.config=/etc/storm/conf/client_jaas.conf -c
storm.thrift.transport=backtype.storm.security.auth.kerberos.KerberosSaslTransportPlugin
```

This command deactivates the topology’s spouts:

```
storm deactivate wordcount12 -c java.security.auth.login.config=/etc/storm/conf/client_jaas.conf -
c storm.thrift.transport=backtype.storm.security.auth.kerberos.KerberosSaslTransportPlugin
```

This command activates the topology’s spouts:

```
storm activate wordcount12 -c java.security.auth.login.config=/etc/storm/conf/client_jaas.conf -c
storm.thrift.transport=backtype.storm.security.auth.kerberos.KerberosSaslTransportPlugin
```

This command rebalances the topology, perhaps to accommodate more nodes. Note that rebalancing may deactivate an active topology for the duration of the message timeout.

storm rebalance wordcount12 -c java.security.auth.login.config=/etc/storm/conf/client_jaas.conf -c storm.thrift.transport=backtype.storm.security.auth.kerberos.KerberosSaslTransportPlugin

Timestamp	Server Type	Client IP	Server IP	DB User Name	SQL Verb	Object Name
2016-07-25 04:10:17	STORM	9.70.156.60	hw-cl5-01.guard.swg.usma.ibm.com	SVORUGA	rebalance	wordcount12
2016-07-25 04:08:38	STORM	9.70.156.60	hw-cl5-01.guard.swg.usma.ibm.com	SVORUGA	activate	wordcount12
2016-07-25 04:07:51	STORM	9.70.156.60	hw-cl5-01.guard.swg.usma.ibm.com	SVORUGA	deactivate	wordcount12
2016-07-25 04:06:30	STORM	9.70.156.60	hw-cl5-01.guard.swg.usma.ibm.com	SVORUGA	killTopology	wordcount15
2016-07-25 03:51:41	STORM	9.70.156.60	hw-cl5-01.guard.swg.usma.ibm.com	SVORUGA	submitTopology	wordcount16

Figure 22. Storm activity in Guardium

Considerations for upgrade

We recommend that you create a detailed plan for managing upgrades of either the Hadoop cluster OS level, Hadoop itself, or Guardium to avoid audit data loss. Because there are Guardium components on the Hadoop cluster, there must be a coordinated communication plan between the Guardium team and the Hadoop administrators whenever an upgrade to either component is required.

Special care must be taken when using the dynamic policy integration for blocking because of the jar files that are manually copied into the component directories.

Upgrading Hadoop

For normal monitoring (that is, no blocking), there should be no impact on the S-TAP. Because Guardium kernel-level module (K-TAP) is not used, there is less sensitivity to the OS kernel than with “regular” S-TAP monitoring with K-TAP. So, unless you are using that same S-TAP to monitor some other data source (not recommended) you can feel free to upgrade your system.

It is recommended to use KTAP_LIVE_UPDATE=Y to allow the S-TAP to install properly even if there is a kernel mismatch. There is no need to restart the S-TAP.

If you are using dynamic policy integration for blocking, an upgrade will change the path for the guardium_evaluator.jar file because of the Hadoop version number change in the usr/hdp path. You must copy the jar file over to the new directories for each component you are monitoring on each node with an S-TAP.

If the jar file is not where Ranger expects it to be, blocking will not work. You will see a message similar to the following in the HDFS namenode log.

2016-11-30 15:22:24,885 ERROR policyevaluator.RangerDefaultPolicyItemEvaluator (RangerDefaultPolicyItemEvaluator.java:newConditionEvaluator(264)) - RangerDefaultPolicyItemEvaluator.newConditionEvaluator(org.

apache.ranger.plugin.conditionevaluator.GuardiumConditionEvaluator): error instantiating evaluator
2016-11-30 15:22:24,885 ERROR policyevaluator.RangerDefaultPolicyItemEvaluator
(RangerDefaultPolicyItemEvaluator.java:init(76)) - RangerDefaultPolicyItemEvaluator(policyId=12): failed to instantiate condition evaluator 'guard-plugin';
evaluatorClassName='org.apache.ranger.plugin.conditionevaluator.GuardiumConditionEvaluator'

Upgrading Guardium S-TAP and the Guardium plug-in

If you upgrade the Guardium S-TAP to a newer level, there should be no impact to existing auditing capabilities. There is no need to restart the server or the Hadoop components.

However, if you are upgrading the Guardium_evaluator.jar file, you must remember to copy that file over to the relevant paths as described in *and* restart the Hadoop components.

Troubleshooting

If you are not seeing traffic from Ranger, check the following:

- Is integration enabled? Check the status on the Hadoop Monitoring configuration UI to see if that appropriate services are green. (or use the grdapi command get_hadoop_cluster_status)
- Did the Hadoop administrator restart the monitored components after the Guardium configuration?
- Are the ports specified in Guardium the as in the log4j configuration?
- Are the Ranger plugins installed for the components you want to monitor? From Ambari, go to **Ambari > Ranger> Ranger plugins**
- Is Ranger auditing is turned on for each component?
- Is Ranger is capturing audits? From the Ranger UI, look at Audits.
- Are the Ranger policies enabled and the right policies are being triggered (especially for blocking)
- Double check your Guardium policy to make sure you are logging the traffic you expect to see from Ranger.

Reference

Guardium APIs

add_ranger_config

Use of this command requires valid administrative authority on the Ambari server such as an admin or service administrator account. After running the command, the Ambari administrator must restart the affected Hadoop components before changes can take effect.

Parameter	**Description**
hostname	Required.

Port	Optional. Default value 8080
userName	Required.
Password	Required.
clusterName	Required. Cluster name as defined to Ambari.

Example:

grdapi add_ranger_config hostname=hw-cl4-05 userName=admin port=8080 password=xxxxx clusterName=Cluster4

Output:

ID=0
Configuration for Cluster: Cluster4 added.

add_ranger_service

Use of this command requires valid administrative authority on the Ambari server such as an admin or service administrator account. After running the command, the Ambari administrator must restart the affected Hadoop components before changes can take effect.

Parameter	**Description**
clusterName	Required. Cluster name as defined to Ambari.
serviceName	Required. List of allowable values: HBASE HDFS HIVE KAFKA STORM
stapHostName	Required.
Port	Optional. Default Value 5555
enableMonitoring	Optional. Indicates whether to enable monitoring for this Hadoop service. Valid values: True or False. Default is False.

Example:

grdapi add_ranger_service clusterName=Cluster4 serviceName=HDFS stapHostName="hw-cl4-01.guard.swg.usma.ibm.com" port=5565 enableMonitoring=false

ID=0
The Hadoop service configuration has been changed. Ask the Hadoop administrator to restart the Hadoop service to activate the changes.
HDFS Monitoring Disabled on hw-cl4-01.guard.swg.usma.ibm.com:5565

disable_monitoring_ranger_service

Use of this command requires valid administrative authority on the Ambari server such as an admin or service administrator account. After running the command, the Ambari administrator must restart the affected Hadoop components before changes can take effect.

Parameter	Description
clusterName	Required. Cluster name as defined to Ambari.
serviceName	Required. List of allowable values: HBASE HDFS HIVE KAFKA STORM

Example:

grdapi disable_monitoring_ranger_service clusterName=Cluster4 serviceName=HDFS
ID=0
The Hadoop service configuration has been changed. Ask the Hadoop administrator to restart the Hadoop service to activate the changes.
HDFS Monitoring Disabled on hw-cl4-01.guard.swg.usma.ibm.com:5565

enable_monitoring_ranger_service

Use of this command requires valid administrative authority on the Ambari server such as an admin or service administrator account. After running the command, the Ambari administrator must restart the affected Hadoop components before changes can take effect.

Parameter	Description
clusterName	Required. Cluster name as defined to Ambari.
serviceName	Required. List of allowable values: HBASE HDFS HIVE KAFKA STORM

Example:

grdapi enable_monitoring_ranger_service clusterName=Cluster4 serviceName=HDFS
ID=0
The Hadoop service configuration has been changed. Ask the Hadoop administrator to restart the Hadoop service to activate the changes.
HDFS Monitoring Disabled on hw-cl4-01.guard.swg.usma.ibm.com:5565

get_hadoop_cluster_status (DEPRECATED) -

This API has been deprecated. Use get_ranger_services_status instead.

Parameter	Description
serverHostName	Required. Ambari host name
serverPort	Required. Ambari port
Username	Required. Admin or Service administrator for Ambari
Password	Required. Password for the cluster
clusterName	Required. The name of the cluster

Example:
grdapi get_hadoop_cluster_status serverHostName=hw-cl5-06.guard.swg.usma.ibm.com serverPort=8080 userName=admin password=admin clusterName=c5
HDFS Monitoring Enabled
HBASE Monitoring Enabled

HIVE Monitoring Enabled
KAFKA Monitoring Enabled

get_ranger_config

Use this to get the, Ambari administrator ID, Ambari server name, and Ambari port for the specified Ambari cluster.

Parameter	Description
clusterName	Required. Cluster name as defined to Ambari.

Example:

grdapi get_ranger_config clusterName=Cluster4

admin@hw-cl4-05:8080 Cluster4

get_ranger_services_status

This API indicates whether monitoring is enabled for a particular Hadoop component using the Ranger integration. It does *not* necessarily indicate that traffic is flowing.

Parameter	Description
clusterName	Required. Cluster name as defined to Ambari.

grdapi get_ranger_services_status clusterName=Cluster4

HDFS Monitoring Disabled on hw-cl4-01.guard.swg.usma.ibm.com:5565

list_ranger_configs

This command lists all Ranger configurations. The result is shown as the administrator account, the host and port, and the Cluster name.

Example:
grdapi list_ranger_configs

Output:
ID=0
admin@hw-cl4-05:8080 Cluster4

list_ranger_staps

This command lists all S-TAPs that are eligible for Ranger configuration or are currently configured for Ranger integration. In other words, all S-TAPS that have *not* been configured for Kafka (Cloudera integration).

Example:
grdapi list_ranger_staps

Output:
ID=0
hw-cl5-01.guard.swg.usma.ibm.com:5556
hw-cl4-01.guard.swg.usma.ibm.com:5555

remove_ranger_config

This command deletes the Ranger configuration for the specified cluster. All monitoring for these services will be disabled upon restart of the affected Hadoop components. Use of this command requires valid administrative authority on the Ambari server such as an admin or service administrator account.

Parameter	Description
clusterName	Required. Cluster name as defined to Ambari.

Example:
grdapi remove_ranger_config clusterName=Cluster4

Output:

ID=0
The Hadoop service has been configured to disable monitoring. Ask the Hadoop administrator to restart the Hadoop services to activate the changes
Configuration with ID: 2 deleted successfully

remove_ranger_service

For the specified cluster, this command removes a service from monitoring. Use of this command requires valid administrative authority on the Ambari server such as an admin or service administrator account. After running the command, the Ambari administrator must restart the affected Hadoop components before changes can take effect.

Parameter	Description
clusterName	Required. Cluster name as defined to Ambari.

serviceName	Required. List of allowable values: HBASE HDFS HIVE KAFKA STORM

Example:
grdapi remove_ranger_service clusterName=Cluster4 serviceName=HDFS

Output:

ID=0
Service: HDFS deleted successfully

update_ranger_config

This command lets you update configuration parameters for the Ranger integration. Use of this command requires valid administrative authority on the Ambari server such as an admin or service administrator account. After running the command, the Ambari administrator must restart the affected Hadoop components before changes can take effect.

Parameter	**Description**
clusterName	Required. Cluster name as defined to Ambari.
newClusterName	Optional.
Hostname	Required.
Port	Required. Default value 8080
userName	Required.
Password	Required.

Example:

grdapi update_ranger_config hostname=hw-cl4-05 userName=admin port=8080 password=*xxxxx* clusterName=Cluster4

ID=0
Configuration for Cluster: Cluster4
Updated.

admin@hw-cl4-05:8080 Cluster4

update_ranger_service

Use of this command requires valid administrative authority on the Ambari server such as an admin or service administrator account. After running the command, the Ambari administrator must restart the affected Hadoop components before changes can take effect.

Parameter	**Description**
clusterName	Required. Cluster name as defined to Ambari.
serviceName	Required. List of allowable values: HBASE HDFS HIVE KAFKA STORM
stapHostName	Required.
Port	Optional. Default Value 5555

Example:

grdapi update_ranger_service clusterName=Cluster4 serviceName=HDFS stapHostName="hw-cl4-01.guard.swg.usma.ibm.com" port=5534

ID=0
The Hadoop service configuration has been changed. Ask the Hadoop administrator to restart the Hadoop service to activate the changes.
HDFS Monitoring Enabled on hw-cl4-01.guard.swg.usma.ibm.com:5534
Note: Only one port can be configured per S-TAP host. Changing the port here will update it automatically for all Hadoop services configured to this S-TAP.

update_stap_config

With this command, you can:

- Enable or disable Ranger integration
- Change the number of connections
- Change the listener port used by S-TAP for Ranger.

Restart the S-TAP after changing any settings.

Parameter	Description
log4j_reader_enabled	Enable log4j listening mode for Ranger traffic. 0=no (the default) 1=yes
log4j_listen_address	This is the address that the Ranger plugins will try to connect to. 0.0.0.0 means any address of this machine. Localhost means only listen on the loopback network device on the machine. The default and recommended value is 0.0.0.0, which enables S-TAP to receive traffic from any host. Use localhost if configuring the system for high availability as described above in Standby deployment options. If you choose to restrict access, be sure you are not restricting access to necessary traffic for monitoring.
log4j_num_connections	Number of concurrent connections to expect from the service or services defined to this S-TAP. The default is 20.

Example:

The following command enables the S-TAP for the Ranger integration and specifies the listen address:

grdapi update_stap_config stapHost=sandbox.hortonworks.com
updateValue=TAP.log4j_reader_enabled:1&TAP.log4j_listen_address:9.70.148.147

Python script to configure Ranger (guard_log4j_config.py)

There is a script to configure Ranger outside of the Guardium user interface. This may be useful, for example, if Ambari is using SSL and the UI cannot connect to Ambari.

The Python script is shipped with the S-TAP in the directory where the S-TAP is running; for example, /usr/local/guardium/guard_stap/guard_log4j_listener_config.py.

Here are the parameters you can pass into the script. Table 6 has explanations for the parameters.

guard_log4j_listener_config.py -a <Ambari UI IP> -b <Ambari UI port> -u <Ambari_admin_username> -p <Ambar_admin_password> -l <Guardium log4j listener port> -c <cluster_name> -s <Hadoop_service> -x <enable|disable|check> <--SSL>

Table 6. Parameters to use with Python script to configure Ranger for Guardium integration

Parameter	Value
-a	Host name or IP address of the Ambari server
-b	Ambari UI port. Default is 8080
-u	Ambari server user name. Must be an admin or service admin user.
-p	Password for the admin user
-l	Guardium log4j listener port. The default is 5555.
-c	The Hadoop cluster name
-s	The Hadoop component (service) on which to enable monitoring. Valid values are: • HDFS • HIVE • HASE • KAFKA Only one service can be entered per execution of the script.
-x	Options include: • Enable – enable monitoring • Disable – disable monitoring • Check – check status

--SSL	Indicates that Ambari is using SSL, so Guardium will attempt the connection using SSL. (Note that there is no support for client certificate authentication using this script.)

Example:

./guard_log4j_listener_config.py -a 192.168.42.78 -b 8080 -u admin -p admin -l 5555 -c Sandbox -s hbase -x enable --SSL

CLI command – store_ranger_config (DEPRECATED)

This command has been deprecated. Use the grdapi commands add_ranger_config and add_ranger_service instead.

Configure Ranger integration. Use of this command requires valid administrative authority on the ambari server such as an admin or service administrator account. After running the command, the Ambari administrator must restart the affected Hadoop components before changes can take effect.

Parameter	Value
Ambari server host name	Host name or IP address of the Ambari server
Server port number	Ambari server port (or leave blank to accept the default 8080)
Ambari server user name	Must be an admin or service admin user.
Password	Password for the admin user
Cluster name	The Hadoop cluster name
Service name	The Hadoop component on which to enable monitoring. Valid values are: • HDFS • HIVE • HASE • KAFKA Only one service can be entered per command.
Do you want to enable service (y/n)	Y will enable the Hadoop component for Guardium monitoring.
Host where S-TAP is installed.	For the Hadoop component entered previously, enter the S-TAP host or IP that should collect the audit events from Ranger.
Listener port number	Enter 5555 or leave blank to default to 5555

Example:

store ranger_config

```
Please enter the following parameters to proceed:
Enter the Ambari server host name: hw-cl5-06b
Enter the server port number:
Enter the Ambari server username: svoruga
Enter the password for svoruga@hw-cl5-06 ? ********
Enter the cluster name: c5
Enter the service name: hdfs
Do you want to enable service? (y/n)y
Enter the host where S-TAP is installed: hw-cl5-01
Enter the listener port number: (press ENTER to use default port):
```

Using REST APIs to configure monitoring

If you need a way to configure monitoring from an external system, you can use the Guardium REST API. This section shows an end to end example of using the Guardium REST API to configure Guardium for monitoring Apache Ranger.

Important: This is not a primer on Guardium REST APIs. For more details on using Guardium REST API, see Using the Guardium REST API on IBM developerWorks.

The example below was run over different days and times. The client secret token does expire periodically and will have to be re-obtained so it is not the same throughout this example.

1. Get the shared secret:

```
grdapi register_oauth_client client_id=client_id1
ID=0
{"client_id":"client_id1","client_secret":"11f463c8-481d-4f09-
863b-
50480dbfaa33","grant_types":"password","scope":"read,write","redi
rect_uri":"https://someApp"}
ok
```

2. Get the access token:

```
curl -k -X POST -d "client_id=client_id1&client_secret=11f463c8-
481d-4f09-863b-
50480dbfaa33&grant_type=password&username=admin&password=1qaz%21Q
AZ" https://javier-vm02:8443/oauth/token
{"access_token":"450c1f26-cc61-4c0d-bf19-
403e1972f826","token_type":"bearer","expires_in":10799,"scope":"r
ead write"}
```

   Use this access token for subsequent API invocations in this session.

3. List the current Ranger configuration (cluster) defined to this Guardium appliance. In this example, there is no configuration yet.

```
curl -k -i --header "Authorization:Bearer 450c1f26-cc61-4c0d-
bf19-403e1972f826" https://javier-
vm02.guard.swg.usma.ibm.com:8443/restAPI/list_ranger_configs
```

Response:

```
HTTP/1.1 200 OK
X-FRAME-OPTIONS: SAMEORIGIN
Set-Cookie: JSESSIONID=8B9581F4DA85E980DBD4C152CB8AD975; Path=/;
Secure; HttpOnly
Cache-Control: max-age=86400
Expires: Tue, 20 Sep 2016 21:32:09 GMT
Access-Control-Allow-Methods: POST, GET,  PUT, DELETE
Access-Control-Allow-Headers: authorization, origin, X-Requested-
With, Content-Type, Accept
Access-Control-Max-Age: 18000
Content-Type: application/json;charset=UTF-8
Content-Length: 62
Date: Mon, 19 Sep 2016 21:32:09 GMT
Server: SQL Guard

{
  "ID": 0,
  "Message": "ID=0 - No configurations found."
}
```

4. Now, add the cluster information (Ranger config). For brevity, the http header information is redacted from the output for the rest of the examples in this section.

```
curl -k --header "Authorization:Bearer f4850460-24e4-492d-b3ef-
23d1b073eb05" -i -H "Content-Type: application/json" -X POST -d
'{hostname="hw-cl4-05", userName="admin", port=8080,
password="admin", clusterName="Cluster4"}' https://javier-
vm02:8443/restAPI/add_ranger_config
```

Response:

```
[
  {
    "id": 2,
    "clusterName": "Cluster4",
    "serverHost": "hw-cl4-05",
    "serverPort": 8080,
    "userName": "admin",
    "password": "admin",
    "lastRefresh": "2016-09-27 11:31:03",
    "status": []
  }
]
```

5. Verify the configuration just added:
   curl -k -i --header "Authorization:Bearer 5c786181-6247-4818-ac77-320dbcf88dd5" https://javier-vm02:8443/restAPI/**get_ranger_config?clusterName=Cluster4**

   Response:


```
[
  {
    "id": 1,
    "clusterName": "Cluster4",
    "serverHost": "hw-cl4-05",
    "serverPort": 8080,
    "userName": "admin",
    "password": "admin",
    "lastRefresh": "2016-10-03 18:06:31",
    "status": []
  }
]
```


6. List the available S-TAPs. You will need to choose one or more S-TAP to use in a later step when configuring the Hadoop services.

   curl -k -i --heade-492d-b3ef-23d1b073eb05" https://javier-vm02:8443/restAPI/**list_ranger_staps**

   Response:


```
 [
  {
    "id": 18,
    "name": "hw-cl4-01.guard.swg.usma.ibm.com",
    "value": "hw-cl4-01.guard.swg.usma.ibm.com",
    "port": "5534",
    "stapStatus": 2
  },
  {
    "id": 24,
    "name": "hw-cl5-01.guard.swg.usma.ibm.com",
    "value": "hw-cl5-01.guard.swg.usma.ibm.com",
    "port": "5534",
    "stapStatus": 2
  },
  {
    "id": 25,
    "name": "mdb01.guard.swg.usma.ibm.com",
    "value": "mdb01.guard.swg.usma.ibm.com",
    "port": "5555",
    "stapStatus": 2
  }
```

```
]
```

Only S-TAPs with stapStatus=2 will be displayed, which means the StAP is active and properly synchronized.

7. Look at the services that are set up to be monitored. In this example, since we have not yet configured any services for monitoring, the Ambari configuration ID is -1 and the ID of the service is 0.

```
 curl -k -i --header "Authorization:Bearer f4850460-24e4-492d-
b3ef-23d1b073eb05" https://javier-
vm02:8443/restAPI/get_ranger_services_status?clusterName=Cluster4
```

Response:

```
[
 {
   "id": 0,
   "ambariConfigId": -1,
   "service": {
     "id": 2,
     "label": "HDFS",
     "value": "HDFS"
   },
   "isMonitored": false,
   "editMode": false
 },
 {
   "id": 0,
   "ambariConfigId": -1,
   "service": {
     "id": 3,
     "label": "Hive",
     "value": "HIVE"
   },
   "isMonitored": false,
   "editMode": false
 },
 {
   "id": 0,
   "ambariConfigId": -1,
   "service": {
     "id": 1,
     "label": "HBase",
     "value": "HBASE"
   },
   "isMonitored": false,
   "editMode": false
 },
 {
   "id": 0,
```

```
    "ambariConfigId": -1,
    "service": {
      "id": 5,
      "label": "Storm",
      "value": "STORM"
    },
    "isMonitored": false,
    "editMode": false
  },
  {
    "id": 0,
    "ambariConfigId": -1,
    "service": {
      "id": 4,
      "label": "Kafka",
      "value": "KAFKA"
    },
    "isMonitored": false,
    "editMode": false
  }
]
```


8. Map the Hadoop services you want to monitor to the desired S-TAP and, optionally, enable monitoring for those services. To do this, you need the S-TAP host and port from the output of get_ranger_staps above. In this example, only HDFS, HIVE, and HBASE are specified. This example does not enable monitoring.

```
curl -k --header "Authorization:Bearer f4850460-24e4-492d-b3ef-
23d1b073eb05" -i -H "Content-Type: application/json" -X POST -d
'{clusterName="Cluster4", serviceName="HDFS,HIVE,HBASE",
stapHostName="hw-cl4-01.guard.swg.usma.ibm.com", port=5534,
enableMonitoring=false}' https://javier-
vm02:8443/restAPI/add_ranger_service
```

Response:


```
 [
  {
    "id": 3,
    "ambariConfigId": 1,
    "service": {
      "id": 1,
      "label": "HBase",
      "value": "HBASE"
    },
    "stapHost": {
      "id": 18,
      "name": "hw-cl4-01.guard.swg.usma.ibm.com",
      "value": "hw-cl4-01.guard.swg.usma.ibm.com",
      "port": "5534",
      "stapStatus": 2
```

```
    },
    "isMonitored": false,
    "port": "5534",
    "editMode": true
  },
  {
    "id": 1,
    "ambariConfigId": 1,
    "service": {
      "id": 2,
      "label": "HDFS",
      "value": "HDFS"
    },
    "stapHost": {
      "id": 18,
      "name": "hw-cl4-01.guard.swg.usma.ibm.com",
      "value": "hw-cl4-01.guard.swg.usma.ibm.com",
      "port": "5534",
      "stapStatus": 2
    },
    "isMonitored": false,
    "port": "5534",
    "editMode": true
  },
  {
    "id": 2,
    "ambariConfigId": 1,
    "service": {
      "id": 3,
      "label": "Hive",
      "value": "HIVE"
    },
    "stapHost": {
      "id": 18,
      "name": "hw-cl4-01.guard.swg.usma.ibm.com",
      "value": "hw-cl4-01.guard.swg.usma.ibm.com",
      "port": "5534",
      "stapStatus": 2
    },
    "isMonitored": false,
    "port": "5534",
    "editMode": true
  }
]
```

9. If you did not enable monitoring on add_ranger_service, you can use the enable_monitoring_ranger_service API. This API operates on only one service at a time. This example is just enabling it for HIVE.

```
 curl -k --header "Authorization:Bearer 2aff7ed2-f3eb-402b-ac84-
e921fa836900" -i -H "Content-Type: application/json" -X PUT -d
```

```
'{clusterName="Cluster4", serviceName="HIVE", stapHostName="hw-
cl4-01.guard.swg.usma.ibm.com", port=5534 }' https://javier-
vm02:8443/restAPI/enable_monitoring_ranger_service
```

Response:


```
[
  {
    "id": 3,
    "ambariConfigId": 1,
    "service": {
      "id": 1,
      "label": "HBase",
      "value": "HBASE"
    },
    "stapHost": {
      "id": 30,
      "name": "hw-cl4-01.guard.swg.usma.ibm.com",
      "value": "hw-cl4-01.guard.swg.usma.ibm.com",
      "port": "5534",
      "stapStatus": 2
    },
    "isMonitored": false,
    "port": "5534",
    "editMode": true
  },
  {
    "id": 1,
    "ambariConfigId": 1,
    "service": {
      "id": 2,
      "label": "HDFS",
      "value": "HDFS"
    },
    "stapHost": {
      "id": 30,
      "name": "hw-cl4-01.guard.swg.usma.ibm.com",
      "value": "hw-cl4-01.guard.swg.usma.ibm.com",
      "port": "5534",
      "stapStatus": 2
    },
    "isMonitored": false,
    "port": "5534",
    "editMode": true
  },
  {
    "id": 2,
    "ambariConfigId": 1,
    "service": {
      "id": 3,
      "label": "Hive",
```

```
      "value": "HIVE"
    },
    "stapHost": {
      "id": 30,
      "name": "hw-cl4-01.guard.swg.usma.ibm.com",
      "value": "hw-cl4-01.guard.swg.usma.ibm.com",
      "port": "5534",
      "stapStatus": 2
    },
    "isMonitored": true,
    "port": "5534",
    "editMode": true
  }
]
```


10. List all configured clusters. The output in this example is what you would see if all services were configured.

```
curl -k -i --header f4850460-24e4-492d-b3ef-23d1b073eb05
https://javier-
vm02.guard.swg.usma.ibm.com:8443/restAPI/list_ranger_configs
```

Response:


```
 [
  {
    "id": 1,
    "clusterName": "Cluster4",
    "serverHost": "hw-cl4-05",
    "serverPort": 8080,
    "userName": "admin",
    "password": "admin",
    "lastRefresh": "2016-09-14 13:45:10",
    "status": [
      {
        "id": 1,
        "ambariConfigId": 1,
        "service": {
          "id": 1,
          "label": "HBase",
          "value": "HBASE"
        },
        "stapHost": {
          "id": 22,
          "name": "hw-cl4-01.guard.swg.usma.ibm.com",
          "value": "hw-cl4-01.guard.swg.usma.ibm.com",
          "port": "5555",
          "stapStatus": 2
        },
        "isMonitored": true,
        "port": "5555",
```

```
  "editMode": true
},
{
  "id": 2,
  "ambariConfigId": 1,
  "service": {
    "id": 2,
    "label": "HDFS",
    "value": "HDFS"
  },
  "stapHost": {
    "id": 22,
    "name": "hw-cl4-01.guard.swg.usma.ibm.com",
    "value": "hw-cl4-01.guard.swg.usma.ibm.com",
    "port": "5555",
    "stapStatus": 2
  },
  "isMonitored": false,
  "port": "5555",
  "editMode": true
},
{
  "id": 3,
  "ambariConfigId": 1,
  "service": {
    "id": 3,
    "label": "Hive",
    "value": "HIVE"
  },
  "stapHost": {
    "id": 28,
    "name": "hw-cl5-01.guard.swg.usma.ibm.com",
    "value": "hw-cl5-01.guard.swg.usma.ibm.com",
    "port": "5534",
    "stapStatus": 2
  },
  "isMonitored": true,
  "port": "5534",
  "editMode": true
},
{
  "id": 4,
  "ambariConfigId": 1,
  "service": {
    "id": 4,
    "label": "Kafka",
    "value": "KAFKA"
  },
  "stapHost": {
    "id": 22,
    "name": "hw-cl4-01.guard.swg.usma.ibm.com",
```

```
            "value": "hw-cl4-01.guard.swg.usma.ibm.com",
            "port": "5555",
            "stapStatus": 2
          },
          "isMonitored": true,
          "port": "5555",
          "editMode": true
        },
        {
          "id": 5,
          "ambariConfigId": 1,
          "service": {
            "id": 5,
            "label": "Storm",
            "value": "STORM"
          },
          "stapHost": {
            "id": 22,
            "name": "hw-cl4-01.guard.swg.usma.ibm.com",
            "value": "hw-cl4-01.guard.swg.usma.ibm.com",
            "port": "5555",
            "stapStatus": 2
          },
          "isMonitored": false,
          "port": "5555",
          "editMode": true
        }
      ]
    }
]
```


GIM Parameter - log4j_reader_enabled

A new Guardium Installation Manager (GIM) parameter, log4j_reader_enabled, can be used to signify that this S-TAP is eligible for the Ranger integration. Additional configuration is required to enable and activate monitoring, as described in this guide.

Resources

- Tutorial on Ranger security policies: http://hortonworks.com/hadoop-tutorial/manage-security-policy-hive-hbase-knox-ranger/
- Hortonworks security guide: http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.4.2/bk_Security_Guide/content/index.html
- Information about Kakfa: http://kafka.apache.org/
- Storm commands (command line): http://storm.apache.org/releases/1.0.1/Command-line-client.html
- Storm tutorial: http://storm.apache.org/releases/current/Tutorial.html

- developerWorks article on Using Guardium REST APIs: http://www.ibm.com/developerworks/data/library/techarticle/dm-1404guardrestapi/index.html

Notices