Security Architecture

Architecture Design Considerations

The design of the MapR security architecture takes into account the main threats to a secure cluster. By default, MapR provides basic authorization functionality and some authentication as follows:

Security Feature	Description
Filesystem permissions	MapR-FS is a POSIX-like file system. You can set user permissions as you would on any other Linux system.
Cluster, volume, and job queue Access Control Lists (ACLs)	You can specify the actions that a given user can perform on each of these cluster elements.
Access Control Expressions (ACEs) for natively stored MapR-DB tables	ACEs control which areas of the tables users or groups can access.
Username/password login authentication to the MapR Control System (MCS) through Pluggable Access Modules (PAM)	You can use any registry that has a PAM module.

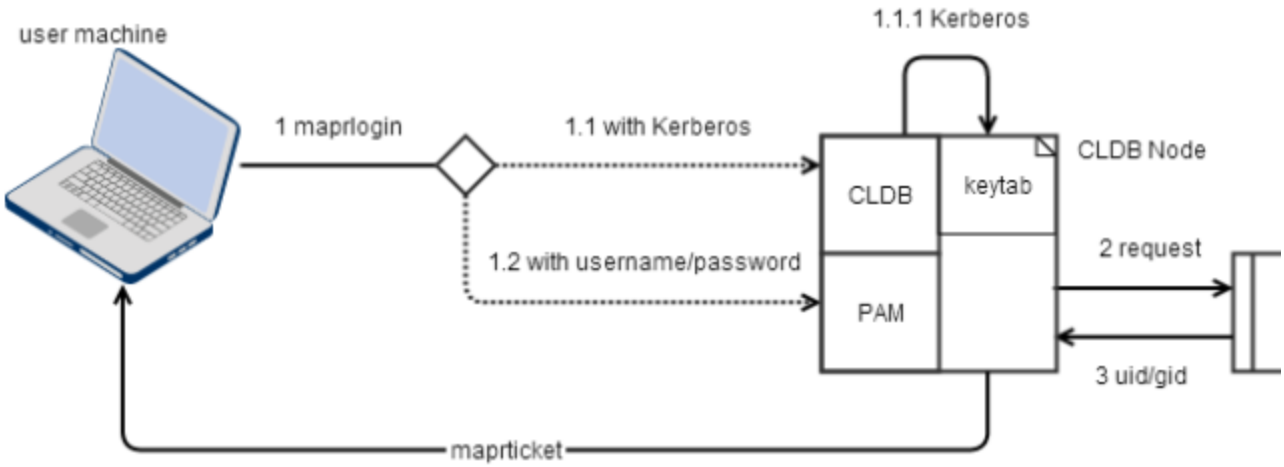
Wire-level security (WLS) is disabled by default. When WLS is enabled, MapR upgrades its security to use network-safe authentication and encryption:

- Communication between the nodes in the cluster is authenticated and may be encrypted:
    - Traffic between the server and cluster, traffic within the MapR filesystem, and CLDB traffic is authenticated using network-safe tokens and may be encrypted with secure MapR RPCs.
    - Traffic between JobClients, TaskTrackers, JobTrackers, NodeManagers, and ResourceManagers is secured with MapR-SASL, an implementation of the Simple Authentication and Security Layer (SASL) framework.
- Support for Kerberos user authentication.
- Support for Kerberos encryption for secure communication to open source components that require it.
- Support for the Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) used with the web UI frontends of some cluster components.

Clusters with different security profiles, and client machines outside of the cluster's security realm, can communicate with the secure cluster.

Authentication Architecture: the maprlogin Utility

The `maprlogin` utility supports user authentication with either username and password or Kerberos to generate a unique session token called a *t icket*. The following diagram outlines the process flow:



Authentication Flow

On clusters that use Kerberos for authentication, a MapR ticket is implicitly obtained for a user that runs a MapR command without first using the `maprlogin` utility. The implicit authentication flow for the `maprlogin` utility first checks for a valid ticket for the user, and uses that ticket if it exists. If a ticket does not exist, the `maprlogin` utility checks if Kerberos is enabled for the cluster, then checks for an existing valid Kerberos identity. When the `maprlogin` utility finds a valid Kerberos identity, it generates a ticket for that Kerberos identity.

When you explicitly generate a ticket, you have the option to authenticate with your username and password or authenticate with Kerberos:

1. The user on the client machine invokes the `maprlogin` utility, which connects to a CLDB node in the cluster using HTTPS. The hostname for the CLDB node is specified in the `mapr-clusters.conf` file.
    a. When using username/password authentication, the node authenticates using PAM modules with the Java Authentication and Authorization Service (JAAS). The JAAS configuration is specified in the `mapr.login.conf` file. The system can use any registry that has a PAM module available.
    b. When using Kerberos to authenticate, the CLDB node verifies the Kerberos principal with the `keytab` file.
2. After authenticating, the CLDB node uses the standard UNIX APIs `getpwnam_r` and `getgrouplist`, which are controlled by the `/etc/nsswitch.conf` file, to determine the user's user ID and group ID.
3. The CLDB node generates a ticket and returns it to the client machine, completing the login communication between the client and the CLDB.
4. After login, the client communicates with a MapR server. The server validates that the ticket is properly encrypted, to verify that the ticket was issued by the cluster's CLDB.
5. The server also verifies that the ticket has not expired or been blacklisted.
6. The server checks the ticket for the presence of a privileged identity such as the `mapr` user. Privileged identities have impersonation functionality enabled.
7. The ticket's user and group information are used for authorization to the cluster, unless impersonation is in effect.

Authorization Architecture: ACLs and ACEs

An Access Control List (ACL) is a list of users or groups. Each user or group in the list is paired with a defined set of permissions that limit the actions that the user or group can perform on the object secured by the ACL. In MapR, the objects secured by ACLs are the job queue, volumes, and the cluster itself.

A job queue ACL controls who can submit jobs to a queue, kill jobs, or modify their priority. A volume-level ACL controls which users and groups have access to that volume, and what actions they may perform, such as mirroring the volume, altering the volume properties, dumping or backing up the volume, or deleting the volume.

An Access Control Expression (ACE) is a combination of user, group, and role definitions. A *role* is a property of a user or group that defines a set of behaviors that the user or group performs regularly. You can use roles to implement your own custom authorization rules. ACEs are used to secure MapR-DB tables that use native storage. See Enabling Table Authorizations with Access Control Expressions.

Encryption Architecture: Wire-Level Security

MapR uses a mix of approaches to secure the core work of the cluster and the Hadoop components installed on the cluster. For example, nodes in a MapR cluster use different protocols depending on their tasks:

- The FileServer, JobTracker, TaskTracker, NodeManager, and ResourceManager use MapR tickets to secure their remote procedure calls (RPCs) with the native MapR security layer. Clients can use the `maprlogin` utility to obtain MapR tickets. Web UI elements of these components use password security by default, but can also be configured to use SPNEGO.
- Hive Metastore, Hue, Flume, and Oozie use MapR tickets by default, but can also be configured to use Kerberos.
- HBase *requires* Kerberos for secure communications.
- The MCS Web UI is secured with passwords. The MCS Web UI does not support SPNEGO for users, but supports both password and SPNEGO security for REST calls.

Servers must use matching security approaches. When an Oozie server, which supports MapR Tickets and Kerberos, connects to HBase, which supports only Kerberos, Oozie must use Kerberos for outbound security. When servers have both MapR and Kerberos credentials, these credentials must map to the same User ID to prevent ambiguity problems.

Security Protocols Used by MapR

For information on specific security protocols supported by different components, see the Security Support Matrix.

Protocol	Encryption	Authentication
MapR RPC	AES/GCM	maprticket
Hadoop RPC and MapR-SASL	AES/GCM	maprticket
Hadoop RPC and Kerberos	Kerberos	Kerberos ticket
Generic HTTP Handler	HTTPS using SSL/TLS	maprticket, username and password, or Kerberos SPNEGO

HTTPS Excluded Ciphers

By default, the following weak TLS/SSL ciphers are excluded from MapR's HTTPS implementation:

- `SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_RSA_EXPORT_WITH_RC4_40_MD5`

You can modify this list of excluded ciphers by editing the `hadoop.ssl.exclude.cipher.suites` property in the `core-site.xml` file. Restart the web servers that use the HTTPS protocol after changing the list of excluded ciphers. The following web servers use HTTPS:

- MCS
- JobTracker
- TaskTracker
- NodeManager
- ResourceManager
- HistoryServer
- CLDB
- HBase