Apache Derby Security

Jean T. Anderson
jta@bristowhill.com
jta@apache.org
Goals
- Understand how to take advantage of Apache Derby security features with a focus on the simplest options and configurations
- Position for understanding and enabling more sophisticated options

Topics
- Apache Derby in a Nut Shell
- User Authentication
- User Authorization
- Database Encryption
- Java 2 Security Manager
Apache Derby in a Nutshell

- Complete relational database
- Implemented in Java
- Standards based (SQL, Java, JDBC)
- Small enough to invisibly embed in an application
- Easy to deploy, use, manage
- Secure

Fully Embeddable or Server-based
Apache Derby in a Nutshell

- Apache DB Subproject
  - http://db.apache.org/derby

- Mail Lists
  - derby-user@db.apache.org
  - derby-dev@db.apache.org

- Wiki
  - http://wiki.apache.org/db-derby/
Where Derby Databases Run

Typical:
- Servers
- Workstations
- Notebooks
- Laptops
- PDAs
- Kiosks
- CD ROMs
- Email inboxes

Not typical:
- Locked machine room
- Highly secured server
- Behind a locked door
Apache Derby Security Strategy

- **User authentication**
  - Restricts access to database(s)

- **User authorization**
  - Restricts access to database objects

- **Database Encryption**
  - Protects physical files

- **Java 2 Security Manager**
  - Takes advantage of Java features
  - Lets you secure the Derby Network Server
User authentication

- Restricts access to database(s)
- Based on a user id and password
- JDBC `user` and `password` attributes in connection URL or properties object
- `user` and `password` parameters in `DriverManager.getConnection()` methods
- `user` and `password` properties in `DataSource`
User authentication: URL syntax

**Embedded:**
```
ij> connect 'jdbc:derby:DbTest;user=app;password=derby';
```

**Derby Network Client:**
```
ij> connect 'jdbc:derby://localhost:1527/DbTest;user=app;password=derby';
```

**IBM DB2 Universal JDBC Driver:**
```
ij> connect 'jdbc:derby:net://localhost:1527/DbTest:user=app;password=derby';
```

The exact syntax depends on the JDBC driver.
User authentication

- Four types
  - NONE (default)
  - LDAP
  - BUILTIN (will demo)
  - Application-defined

- Granularity
  - Per database (set as database properties)
  - For the system (derby.properties file)
User authentication: NONE

- This is the default
  - Gets developers up and running quickly, easily
- No user name required to connect
  - Defaults to APP (default schema is also APP)
- No password required to connect
- If user and password are supplied…
  - Schema defaults to the specified user
  - Schema for that user doesn’t need to exist
  - Password is ignored
User authentication: LDAP

- **Properties**
  
  - `derby.connection.requireAuthentication=true`
  - `derby.authentication.provider=LDAP`
  - `derby.authentication.server=ldap_server:389`
  
  - plus optional properties

- **Caveats**
  
  - Derby does not cache LDAP entries
  - Derby does not support LDAP groups
User authentication: App-defined

- **Properties**
  - `derby.connection.requireAuthentication=true`
  - `derby.authentication.provider=java_class_name`

- **Java class implements**
  - `org.apache.derby.authentication.UserAuthenticator`

- **authenticateUser() method**
  - Takes connection info
  - Returns
    - True: user successfully authenticated
    - False: user failed authentication
User authentication: BUILTIN

- **Properties**
  
  ```
  derby.connection.requireAuthentication=true
  derby.authentication.provider=BUILTIN
  ```

- **User-defined using properties**
  
  ```
  derby.user.name=password
  ```

- **System-level: add to derby.properties file**
  
  ```
  derby.user.foo=bar
  ```

- **Database-level:**
  
  ```
  CALL SYSCS_UTIL.SYSCS_SET_DATABASE_PROPERTY ('derby.user.foo', 'bar')
  ```
  
  - Password is stored encrypted internally (sha-1)

Don’t forget to set these properties.
Time out: Derby properties

- System-wide properties
  - Apply to all databases within a system
  - May be set programmatically
  - May be set via an `derby.properties` file

- Database properties
  - Valid only for that database
  - Set via stored procedures
  - Stored in the database

- Can disable system-level properties, but test first 😊

```sql
CALL SYSCS_UTIL.SYSCS_SET_DATABASE_PROPERTY('derby.database.propertiesOnly', 'true')
```
User authentication: Password encryption

- Across the wire userid/password encryption
- Derby Network Server currently supports
  - EUSRIDPWD scheme
  - IBM JCE 1.2.1
- Coming: DERBY-528
  - Strong user id and password substitute authentication using USRSSBPWD scheme
  - Opens to any JCE provider
  - Early testing with Bouncy Castle JCE
User authentication: Demo

$ cat derby.properties
derby.connection.requireAuthentication=true
derby.authentication.provider=BUILTIN
derby.user.jta=foxhound

$ java org.apache.derby.tools.ij
ij> connect 'jdbc:derby:MyDbTest';
ERROR 08004: Connection refused : Invalid authentication.

ij> connect 'jdbc:derby:MyDbTest;user=jta;password=foxhound';
ij> create table dogs (name varchar(15), breed varchar(15));
0 rows inserted/updated/deleted
ij> insert into dogs values ('Shelby', 'foxhound');
1 row inserted/updated/deleted
User authentication to authorization id

- User authentication
  - Case sensitive (likely)
    ```java
    ij> connect 'jdbc:derby:DbTest;user=Mickey;password=Mouse';
    ```

- Database user authorization id
  - Case insensitive: **MICKEY**
  - Unless quoted:
    ```java
    ij> connect 'jdbc:derby:DbTest;user="Mickey";password=Mouse';
    ```
User authorization

- Restricts access to database objects
- Three options
  - \texttt{fullAccess}: Read & modify data (default)
  - \texttt{readOnlyAccess}: Read-only
  - \texttt{noAccess}: Cannot connect
- Granularity
  - Per database (set as database properties)
  - For the system (derby.properties file)
- Coming: Grant/Revoke (DERBY-464)
User authorization

- Properties
  - `derby.database.defaultConnectionMode`
    - fullAccess, readOnlyAccess, noAccess
  - `derby.database.fullAccessUsers`
  - `derby.database.readOnlyAccessUsers`

- Database-level examples

```sql
CALL SYSCS_UTIL.SYSCS_SET_DATABASE_PROPERTY ('derby.database.defaultConnectionMode', 'noAccess')
CALL SYSCS_UTIL.SYSCS_SET_DATABASE_PROPERTY ('derby.database.readOnlyAccessUsers', 'mary,guest')
CALL SYSCS_UTIL.SYSCS_SET_DATABASE_PROPERTY ('derby.database.fullAccessUsers', 'sa')
```
User authorization: Demo

- ‘jta’ has full access to any database in this system
- ‘shelby’ has read only access
- Everybody else has no access

```
$ cat derby.properties
derby.connection.requireAuthentication=true
derby.authentication.provider=BUILTIN
derby.user.jta=foxhound
derby.user.shelby=people
derby.user.squirrel=lettuce
derby.database.defaultConnectionMode=noAccess
derby.database.fullAccessUsers=jta
derby.database.readOnlyAccessUsers=shelby
```
User authorization: Demo

'jta' ability:
ij> connect
  'jdbc:derby:MyDbTest;user=jta;password=foxhound';
ij> select * from dogs;
NAME       | BREED
--------------------
Shelby     | foxhound
ij> insert into dogs values
  ('Ursi', 'mutt'), ('Eliza','mutt');
2 rows inserted/updated/deleted
User authorization: Demo

‘s sh elby’ ability:
ij> connect
   'jdbc:derby:MyDbTest;user=shelby;password=people';
ij> select * from jta.dogs;
NAME   | BREED
---------------------
Shelby   | foxhound
Ursi     | mutt
Eliza    | mutt
3 rows selected

ij> insert into jta.dogs values ('Tucker','spaniel');
ERROR 25502: An SQL data change is not permitted for a read-only connection, user or database.

ij> create table my_dogs (name varchar(15), breed varchar(15));
ERROR 25503: DDL is not permitted for a read-only connection, user or database.
User authorization: Demo

‘squirrel’ (default) ability:

ij> connect 'jdbc:derby:MyDbTest;user=squirrel;password=lettuce';
ERROR 04501: Database connection refused.

Error is different than for a user not in derby.properties:

ij> connect 'jdbc:derby:MyDbTest';
ERROR 08004: Connection refused : Invalid authentication.

ij> connect 'jdbc:derby:MyDbTest;user=ursi;password=more_treats';
ERROR 08004: Connection refused : Invalid authentication.
‘jta’ can grant readOnlyAccess for this one database in the system:

```
ij> CALL SYSCS_UTIL.SYSCS_SET_DATABASE_PROPERTY
    ('derby.database.readOnlyAccessUsers',
     'shelby,squirrel');
0 rows inserted/updated/deleted
```

Now ‘squirrel’ can access this one database in the system:

```
ij> connect
    'jdbc:derby:MyDbTest;user=squirrel;password=lettuce';
ij> select * from jta.dogs;
NAME          | BREED
------------- |-------
Shelby        | foxhound
Ursi          | mutt
Eliza         | mutt
```
Database encryption

- Protects physical files
- Complete encryption of on-disk data
  - Indexes and tables
  - Transaction log file
  - Temporary files (for ORDER BY, etc.)
- Includes application and system data
  - Table data
  - System catalog/metadata information
Database encryption

_Not Encrypted:_

- Data in-memory
  - Page cache contents
  - ResultSets
- service.properties
  - Contains minimal info to boot database
    - Can contain some encryption-related info
  - Jar files stored via sqlj.install_jar
  - derby.log error log
I/O Based Encryption
- Data encrypted just before write call to disk
- Data decrypted immediately after read from disk
- Most of the system is unaware
Database encryption

- Derby provides the pluggable framework
- *You* provide
  - Java Cryptographic Extension (JCE) 1.2.1 or higher
    - Optional in J2SE 1.3
    - Included in J2SE 1.4
  - Encryption provider
    - Sun and IBM JVMs include a provider
    - Can use third party provider
      - Sun site lists five provider implementations
    - [http://java.sun.com/jce](http://java.sun.com/jce)
Database encryption: Database Create

- Database configured for encryption at create
  - Remains encrypted with same key forever

- Two modes
  - Database key store
    - Derby generates encryption key
    - Encryption key stored in `service.properties` file
  - External key store
    - Application provides encryption key
    - App uses external key store, such as a smart card
Database encryption: Database Create

- **Connection URL attributes**
  - dataEncryption=true
  - bootPassword=value

- **Default encryption provider**
  - JRE determines default
  - Can specify alternate with encryptionProvider

- **Default encryption algorithm**
  - DES
  - Can specify alternate with encryptionAlgorithm
Database encryption: Booting

- First connection must provide the boot password (database key store) or encryption key (external key store)
- Once database is booted ...
  - Subsequent connection requests can be made without boot password/encryption key
  - Connection requests are subject to authentication and authorization
  - Database remains booted after first connection disconnects
Database encryption: Demo

- DES Key Length = 56 bits
- Boot password length >= key length
- 8 character minimum required by Derby

```sql
ij> connect
   'jdbc:derby:DbTest;create=true;dataEncryption=true;bootPassword=aPach31sMyL1f3';
```

Encryption entries in `service.properties`:
```
dataEncryption=true
encryptionAlgorithm=DES/CBC/NoPadding
derby.encryptionBlockSize=8
encryptionKeyLength=56-8
encryptedBootPassword=a7922fc4eabaddf5-17981
```

Tip: Easily switched to AES with `encryptionProvider=AES/CBC/NoPadding`
Java 2 Security Manager

- Derby supports environments that enable Java 2 Security Manager
- Requires granting specific Java permissions to the Derby code (*next slide*)
- Derby requires only the minimum permissions needed to perform its intended function as a database engine
Java 2 Security Manager

Derby Security Permissions (derby.jar)

- Create class loaders – SQL queries are compiled to byte code and loaded by an internal class loader [Required]
- Read/write permissions for data files [Required]
- Read derby.* system properties
- Read permission on derby.properties
- Read/write permission on derby.log
- Install JCE provider
Java 2 Security Manager: SQL Routines

- SQL Functions and Procedures must
  - Execute controlled actions using privileged blocks
  - Have permission for action granted to their code base (jar file)
    - Currently not possible for jar files stored in db
- The generated class that executes the SQL statement from which they are called has no permissions and will be in the calling stack of the routine
- So, this procedure fails with a security exception:
  ```sql
  CREATE PROCEDURE SHUT_REMOTE_SYSTEM (e int)
  ...
  CALL SHUT_REMOTE_SYSTEM(-1);
  ```
Grant permission to run Derby and access all databases under the Derby system home

```java
grant codeBase "file:c:/db-derby-10.1.2.1-bin/lib/derby.jar" {
    permission java.lang.RuntimePermission "createClassLoader";
    permission java.util.PropertyPermission "derby.*", "read";
    permission java.io.FilePermission "${derby.system.home}/**", "read";
    permission java.io.FilePermission "${derby.system.home}/${/}-", "read,write,delete";
};
```

How to use the policy with a Java application:

```bash
java -Djava.security.manager -Djava.security.policy=full_path -Dderby.system.home=full_path MyJavaApp
```
Java 2 Security Manager: Demo #2

- Secure the Network Server

- Policy file

```java
// Permissions for starting and using Network Server
grant codeBase "file:/Apache/db-derby-10.1.2.1-bin/lib/-" {
    permission java.io.FilePermission "${derby.system.home}", "read";
    permission java.io.FilePermission "${derby.system.home}${/}-", "read, write, delete";
    permission java.io.FilePermission "${user.dir}${/}-", "read, write, delete";
    permission java.util.PropertyPermission "derby.*", "read";
    permission java.util.PropertyPermission "user.dir", "read";
    permission java.lang.RuntimePermission "createClassLoader";
    permission java.net.SocketPermission "localhost", "accept";
};
(continued on the next page ... )
```
(continued from previous page ...)

// Permissions for stopping the Network Server
grant codeBase "file:c:/Apache/db-derby-10.1.2.1-bin/lib/-" {
permission java.net.SocketPermission "localhost", "accept, connect, resolve";
permission java.net.SocketPermission "127.0.0.1", "accept, connect, resolve";
permission java.net.SocketPermission "localhost:*", "accept, connect, resolve";
};

- **Start Network Server**

C:\Apache\db-derby-10.1.2.1-bin>java
-Djava.security.manager
-Djava.security.policy=C:/nsrv.policy
org.apache.derby.drda.NetworkServerControl start
Server is ready to accept connections on port 1527.
Database create in the default system home works:

ij> connect 'jdbc:derby://localhost:1527/MyDbTest;create=true';

Database create a database in another location fails:

ij> connect 'jdbc:derby://localhost:1527//BadTst;create=true';
ERROR XJ040: DERBY SQL error: SQLCODE: -1, SQLSTATE: XJ040, SQLERRMC: Failed to start database '/BadTst', see the next exception for details.::SQLSTATE: XJ001Java exception: 'access denied (java.io.FilePermission C:\BadTst\service.properties read): java.security.AccessControlException'.

Java 2 Security Manager

More Information:

- Authentication, authorization, encryption:
  - *Derby Developer’s Guide*

- Securing the Network Server:
  - *Server and Administrator’s Guide*

- derby-user@db.apache.org

- [http://java.sun.com/jce](http://java.sun.com/jce)

- [http://java.sun.com/security](http://java.sun.com/security)

- [http://java.sun.com/jndi](http://java.sun.com/jndi)
Questions?

- Apache Derby in a Nut Shell
- User Authentication
- User Authorization
- Database Encryption
- Java 2 Security Manager