Apache Shiro - Executive Summary

- Apache Shiro is a powerful, easy-to-use Java Security Framework with a goal to be more powerful and easier to use than the standard Java APIs.
- If you have any interest in Security or need to add security features to you Java program, then you should view this presentation

Apache Shiro

A Powerful and easy-to-use Java Security Framework

By Christopher Lynch

Christopher Lynch - CSCI 5448 Graduate Presentation

History - Origin

- Originally developed as JSecurity in 2004
  - By Les Hazlewood and Jeremy Haile
    - "They could not find a suitable Java Security Framework that operated well at the application level and was frustrated with JAAS (Java Authentication and Authorization Service)".
      - Quoted from http://en.wikipedia.org/wiki/Java_Authentication_and_Authorization_Service

History – Name Changes

- JSecurity was submitted to the Apache Software Foundation in 2008 and renamed Ki (pronounced key) due to trademark concerns.
- However, additional trademark concerns caused it to be renamed again to Shiro which is the Japanese word for Castle.
  - I found some archived emails that showed Les Hazlewood nominated the name Shiro, and there was eventually some votes between Shiro and Aseca (Application SECurity Api) and Shiro won.

History - Version

- Apache Shiro 1.0 was released in July 2010.
- Version 1.1 was released in November 2010.
- Version 1.2 was released in January 2012.
- The current version is 1.2.1 which was released in July 2012.

Primary Concepts

- A powerful, easy-to-use Java Security framework that performs:
  - Authentication
  - Authorization
  - Cryptography
  - Session Management
- Can be used to secure any application:
  - From Command-Line Applications, Mobile Applications to the largest web and enterprise applications.

Primary Concepts

- There are 3 main concepts in the Shiro Architecture:
  - Subject
  - Security Manager
  - Realms

Primary Concepts - Subject

- Subject
  - Refers to the current user.
  - Who is the current user?
  - Or, is the current user allowed to perform the requested action.
- To acquire the Subject using Shiro:

```
import org.apache.shiro.subject.Subject;
import org.apache.shiro.SecurityUtils;
...
Subject currentUser = SecurityUtils.getSubject();
```

Primary Concepts – Security Manager

- Manages security operations for all users.
- The heart of Shiro's architecture.
- An 'Umbrella' object that references many internally nested security components that form an object graph.
- Once the security manager and its internal object graph is configured, it is usually left alone and the application developer spends almost all of their time with the Subject API.

Primary Concepts - Realms

- Acts as a bridge or connector between Shiro and your application's security data.
- A security-specific Data Access Object which encapsulates connection details for data sources and makes the association data available to Shiro as needed.
- When configuring Shiro, you must specify at least one Realm to use for authentication and/or authorization.

Primary Concepts - Authentication

- Authentication is the process of verifying a user's identity.
- This is typically done in 3 steps
  1. Collect the user's identifying information (called principals) and supporting proof of identity (called credentials).
  2. Submit the principals and credentials to the system.
  3. If the submitted credentials match what the system expects for that user identity, the user is considered authenticated; otherwise, the user is not authenticated.

Primary Concepts - Authentication

- Code Example:

```
//1. Acquire submitted principals and
credentials.
AuthenticationToken token =
      new UsernamePasswordToken( user, pass);

//2. Get the current subject.
Subject currentUser =
                  SecurityUtils.getSubject();

//3. Login:
currentUser.login( token );
```

Primary Concepts - Authentication

- Handling Failed Login:

```
try {
    currentUser.login( token );
} catch(IncorrectCredentialsException e) {
    …
} catch( LockedAccountException lae ) {
    …
} catch( AuthenticationException ae ) {
    …
}
```

Primary Concepts - Authorization

- Access Control
  - Controlling what the user can access in the application, such as resources, web pages, etc.

- Code Example:

```
if( subject.hasRole("administrator") ) {
  // Show the 'create user' button
} else {
  // Grey-out the button
}
```

Primary Concepts - Authorization

- Can rewrite previous example:

```
if( subject.isPermitted("user:create")) {
  // Show the 'create user' button
} else {
  // Grey-out the button
}
```

Primary Concepts – Session Management

- Shiro enables a Session programming paradigm for any application.
- Shiro's sessions are container-independent
- Shiro allows for pluggable Session data stores.
  - Such as enterprise caches, relational databases, NoSQL systems, etc.
  - You can configure session clustering once, and it will work the same way regardless of your deployment environment.

Primary Concepts – Session Management

- Code example: Can simply obtain the session from the subject:

```
Session session = subject.getSession();
session.getAttribute( “key”, someValue );
Date start = session.getStartTimestamp();
Date timestamp =
                session.getLastAccessTime();
session.setTimeout( millis );
…
```

Primary Concepts - Cryptography

- Shiro's goal in cryptography is to simplify and make usable the JDK's cryptography support.
- Cryptography is not specific to Subjects in general, so Shiro's API for cryptography is not Subject specific.
- Shiro focuses on cryptographic hashes (aka message digests) and cryptic ciphers.

Primary Concepts - Cryptography

- Cryptographic Hashing – JDK method
  - JDK uses the MessageDigest for cryptographic hashing
  - It has an awkward static method factory-based API instead of an object-oriented one.
  - Your are forced to catch checked exceptions that may never need to be caught.
  - There is no support for hex-encode or Base64-encode message digest output.

Primary Concepts - Cryptography

- Cryptographic Hashing – Shiro method
  - Shiro has simplified the API so that it is much easier to use.
  - Shiro has added support for hex-encode and Base64-encode message digest output.

Primary Concepts - Cryptography

- Cryptographic Hashing – Code example without Shiro:

```
try {
  MessageDigest md =
        MessageDigest.getInstance("MD5");
  md.digest( bytes );
  byte[] hashed = md.digest();
} catch( NoSuchAlgorithmException e ) {
  //Handle Error.
}
```

Primary Concepts - Cryptography

- Cryptographic Hashing – Code example without Shiro:

```
String hex = new Md5Hash(myFile).toHex();
```

- SHA-512 hashing and Base64-encoding of passwords is just as easy:

```
String encodedPassword =
      new Sha512Hash( password, salt,
                      count ).toBase64();
```

Primary Concepts - Cryptography

- Cryptic Ciphers
  - Ciphers are cryptographic algorithms that can reversibly transform data using a key.
  - Shiro introduces CipherService API: a simple, stateless, thread-safe, that can encrypt or decrypt data in one method call.
  - Shiro supports both byte-array encryption/decryption as well as stream-based encryption/decryption.

Primary Concepts - Cryptography

- Cryptic Ciphers
  - Sample code:

```
AesCipherService cipherService =
                        new AesCipherService();
cipherService.setKeySize(256);
//create a test key:
byte[] testKey =
              cipherService.generateNewKey();
//encrypt a file's bytes:
byte[] encrypted =
             cipherService.encrypt( fileByte,
                                    testKey);
```

Primary Concepts – Web Support

- Shiro ships with a robust web support module.
- Only need to define a Shiro Servlet Filter in web.xml:

```
<filter>
    <filter-name>ShiroFilter</filter-name>
    <filter-class>
        org.apache.shiro.web.servlet.IniShiroFilter
    </filter-class>
    <!--no init-param means load the INI config from
        classpath:shiro.ini -->
</filter>
<filter-mapping>
    <filter-name>ShiroFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Primary Concepts – Web Support

- web.xml example:
  - This filter can read shiro.ini to get a consistent configuration experience regardless of the deployment environment.
  - Once configured, the Shiro Filter will filter every request and ensure the request-specific Subject is accessible during the request.
  - You can perform security-specific logic to ensure only requests that meet certain criteria are allowed through.

Primary Concepts – Web Support

- Shiro allows you to specify ad-hoc filter chains for any matching URL pattern.
- This gives you a great deal of flexibility in enforcing security rules.

Primary Concepts – Web Support

- URL-Specific Filter Chains:
  - Just need to configure in shiro.ini:

```
[urls]
/assets/** = anon
/user/signup = anon
/user/** = user
/rpc/reset/** = perms[rpc:invoke], authc
/** = authc
```

  - The values on the left of the '=' represent a context-relative web application path.
  - The values on the right define a filter chain.

Primary Concepts – Web Support

- URL-Specific Filter Chains
  - A filter chain is an ordered, comma delimited list of Servlet filters to execute for the given path.
  - Each filter is a normal Servlet Filter.
  - anon, user, perms and authc are special security-related filters that Shiro provides out-of-the-box.
  - You can use any combination of the security filters including specifying your own.

Primary Concepts – Web Support

- JSP Tag Library
  - This allows you to control the output of your JSP pages based on the current Subject's state.

```
<%@ taglib prefix="shiro"
    uri="http://shiro.apache.org/tags" %>
...
<p>Hello
<shiro:user>
   <!-- shiro:principal prints out the Subject's
        main principal - in this case, a username:
     -->
   <shiro:principal/>!
</shiro:user>
<shiro:guest>
   <!-- not logged in - considered a guest. Show the register
        link: -->
   ! <a href="register.jsp">Register today!</a>
</shiro:guest>
</p>
```

Primary Concepts – Web Support

- JSP Tag Library
  - There are other tags that allow you to include output based on what roles exist or don't exist, what permissions are assigned or not, and whether they are authenticated, remembered (from "Remember Me" services), or an anonymous guest.

Primary Concepts – Limitations

- Shiro does not currently deal with Virtual Machine level security.
  - For example, cannot prevent certain classes from loading in a class loader based on an access control policy.
- Shiro does not currently natively support "multi-stage" authentication, where a user might login via one mechanism, only to be asked to login again via a different mechanism.
  - A real possibility that this will be supported in the future.

Primary Concepts – Limitations

- Currently, all Realm implementations support "read" operations for acquiring authentication and authorization data to perform logins and access control. "Write" operations, like creating user accounts, are not supported.
  - This is because the data model to support these operations varies dramatically across applications, and it would be difficult to enforce a "write" API on all Shiro users.

Conclusion

- Shiro powerful and easy-to-use Java Security framework that provides APIs for
  - Authentication
  - Authorization
  - Sessions
  - Cryptography
  - Web Support