阿男的网络服务器架构指南

Weinan Li

Contents

Chapter 1

介绍

在这个专栏，

JBoss Web Server is a product produced by Red Hat:

https://www.redhat.com/en/technologies/jboss−middleware/web−server

The product is comprised of fully open−source projects coming from Apache and JBoss communities to provide a lightweight solution on establishing web server clusters, such as: Apache Tomcat, HTTPD, mod_cluster, openssl, etc.

And I’d like to share my thoughts on this product while I’m working on it. For example, I’ll introduce the usages of Tomcat, HTTPD, openssl and mod_clusters and how they interactive with each other.

I have written a book in before to introduce part of the area:

https://www.packtpub.com/application−development/jboss−eap6−high−availability

And in future I’d like to keep sharing clustering and SSL related topics here.

Chapter 2

How To Use Tomcat Vault

Tomcat Vault is a tool that allows you to encrypt the passwords in Apache Tomcat configuration files.

For example, here is one line excerpted from “tomcat-users.xml”:

```
<user username="tomcat" password="tomcat" roles="tomcat"/>
```

As we can see above, the password is stored as plaintext and it’s a security risk. Though the configuration in store on server, it’s still very dangerous to store password in such way.

Tomcat Vault is created to solve this problem, it will encrypt your password and store it in standard Java keystore, and let tomcat access the password in a safe way. In this article, I’d like to show you how to use it with Tomcat.

Installation of Apache Tomcat and Tomcat Vault

First we need to have Apache Tomcat and Tomcat-Vault installed on our machine.

For Tomcat, I am using 8.0.39 for this article.

For Tomcat Vault, I just clone the project from GitHub into my local machine and build it from master branch:

```
git clone https://github.com/picketbox/tomcat-vault.git
```

And then using Maven to build and install it:

```
tb13:tomcat-vault weli$ pwd
/Users/weli/projs/tomcat-vault
tb13:tomcat-vault weli$ mvn install
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Vault extension for Apache Tomcat 1.0.8.Final
[INFO] ------------------------------------------------------------------------
[INFO]
...
copy dependency file to the correct module directory:
     [copy] Copying 1 file to
/Users/weli/projs/tomcat-vault/modules/system/layers/base/tomcat-vault/main
[INFO] Executed tasks
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ tomcat-vault ---
[INFO] Installing
/Users/weli/projs/tomcat-vault/target/tomcat-vault-1.0.8.Final.jar to
/Users/weli/.m2/repository/org/apache/tomcat/tomcat-vault/1.0.8.Final/tomcat-vau
lt-1.0.8.Final.jar
[INFO] Installing /Users/weli/projs/tomcat-vault/pom.xml to
/Users/weli/.m2/repository/org/apache/tomcat/tomcat-vault/1.0.8.Final/tomcat-vau
lt-1.0.8.Final.pom
[INFO] Installing
/Users/weli/projs/tomcat-vault/target/tomcat-vault-1.0.8.Final-jar-with-dependen
cies.jar to
/Users/weli/.m2/repository/org/apache/tomcat/tomcat-vault/1.0.8.Final/tomcat-vau
lt-1.0.8.Final-jar-with-dependencies.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2.567 s
```

```
[INFO] Finished at: 2016-12-20T16:34:53+08:00
[INFO] Final Memory: 17M/265M
[INFO] ------------------------------------------------------------------------
```

After building it, we can get the tomcat–vault jars:

```
tb13:tomcat-vault weli$ ls -1 target/*.jar
target/tomcat-vault-1.0.8.Final-jar-with-dependencies.jar
target/tomcat-vault-1.0.8.Final.jar
```

Next we will can try to play with ‘tomcat–vault–1.0.8.Final–jar–with–dependencies.jar’ to see if it can work correctly.

First we should make sure that we are in the ‘target’ directory which contains the generated jar files:

```
tb13:target weli$ pwd
/Users/weli/projs/tomcat-vault/target
tb13:target weli$ ls *.jar
tomcat-vault-1.0.8.Final-jar-with-dependencies.jar tomcat-vault-1.0.8.Final.jar
```

Because the jar file contains a Main class, so we can invoke it like this:

```
tb13:target weli$ java -classpath
tomcat-vault-1.0.8.Final-jar-with-dependencies.jar
org.apache.tomcat.vault.VaultTool
**********************************
****  JBoss Vault  ***************
**********************************
Please enter a Digit::   0: Start Interactive Session  1: Remove Interactive
Session  2: Exit
```

If everything goes fine, you can directly using the java command as shown above to start the org.apache.tomcat.vault.VaultTool.

The next step is to put tomcat–vault jar into our local Apache Tomcat directory:

```
tb13:lib weli$ pwd
/Users/weli/projs/apache-tomcat-8.0.39/lib
tb13:lib weli$ cp
~/projs/tomcat-vault/target/tomcat-vault-1.0.8.Final-jar-with-dependencies.jar .
```

As the command shown above, we have the tomcat–vault jar with dependecies copied into tomcat lib directory.

Till now, the installation step is finished, and next we can start to integrate tomcat–vault with tomcat.

Generating Java Keystore for Tomcat Vault

Tomcat Vault relies on Java Keystore to store the passwords, so the first step is to use keytool command provided by JDK to generate a keystore.

Here is the command to generate keystore:

```
tb13:conf weli$ keytool -genseckey -keystore vault.keystore -alias my_vault
-storetype jceks -keyalg AES -keysize 128 -storepass my_password123 -keypass
my_password123 -validity 730
```

As the command shown above, we have generated a keystore named vault.keystore, and set the password of the store to my_password123. We also generate a key pair with the alias name my_vault, and set the password of this generated key pair to my_password123 (You should use different password for key store and key pair in production environment).

Please note that I have put the above generated keystore file to conf directory of Tomcat:

```
tb13:conf weli$ pwd
/Users/weli/projs/apache-tomcat-8.0.39/conf
tb13:conf weli$ ls vault.keystore
vault.keystore
```

In production environment, you should put the keystore into a safer place and set the permission of the file properly.

Now we can check this keystore by using the keytool command:

```
tb13:conf weli$ keytool -list -v -keystore vault.keystore -storetype jceks
-storepass my_password123

Keystore type: JCEKS
Keystore provider: SunJCE

Your keystore contains 1 entry

Alias name: my_vault
Creation date: Dec 20, 2016
Entry type: SecretKeyEntry


*******************************************
*******************************************
```

As the command output shown above, we can see our keystore contains one SecretKeyEntry named my_vault.

Till now, we have generated the keystore for tomcat vault to use. The next step is to invoke tomcat vault to initialize the keystore for us.

Initializing Tomcat Vault

Now we can invoke tomcat vault to initialize the keystore so it can be used to store tomcat username and password information.

First we need to go to the lib directory of tomcat, because in previous steps we have put tomcat–vault jar into it:

```
tb13:lib weli$ pwd
/Users/weli/projs/apache-tomcat-8.0.39/lib
tb13:lib weli$ ls tomcat-vault*
tomcat-vault-1.0.8.Final-jar-with-dependencies.jar
```

We need to invoke the above tomcat–vault jar to initialize the keystore we generated in previous step, which is named vault.keystore. Here is the whole step to use tomcat–vault to initialize the keystore:

```
tb13:lib weli$ java -classpath
tomcat-vault-1.0.8.Final-jar-with-dependencies.jar
org.apache.tomcat.vault.VaultTool
```

```
**********************************
****  JBoss Vault  ***************
**********************************
Please enter a Digit::   0: Start Interactive Session  1: Remove Interactive
Session  2: Exit
0
Starting an interactive session
Enter directory to store encrypted
files:/Users/weli/projs/apache-tomcat-8.0.39/conf
Enter Keystore URL:/Users/weli/projs/apache-tomcat-8.0.39/conf/vault.keystore
Enter Keystore password:
Enter Keystore password again:
Values match
Enter 8 character salt:1234abcd
Enter iteration count as a number (Eg: 44):120
Enter Keystore Alias:my_vault
Initializing Vault
Dec 21, 2016 3:43:33 PM
org.apache.tomcat.vault.security.vault.PicketBoxSecurityVault init
INFO: PBOX000361: Default Security Vault Implementation Initialized and Ready
Vault Configuration in tomcat properties file:
********************************************
...
KEYSTORE_URL=/Users/weli/projs/apache-tomcat-8.0.39/conf/vault.keystore
KEYSTORE_PASSWORD=MASK-3CuP21KMHn7G6iH/A3YpM/
KEYSTORE_ALIAS=my_vault
SALT=1234abcd
ITERATION_COUNT=120
ENC_FILE_DIR=/Users/weli/projs/apache-tomcat-8.0.39/conf/
...
********************************************
Vault is initialized and ready for use
Handshake with Vault complete
Please enter a Digit::   0: Store a secured attribute  1: Check whether a
secured attribute exists  2: Exit
```

```
2
```

Above is the whole step to use tomcat–vault to initialize the keystore. Let’ s see the above process step by step. Here is the explanation:

```
Please enter a Digit::   0: Start Interactive Session  1: Remove Interactive
Session  2: Exit
0
Starting an interactive session
```

We choose 0 to start an interactive session to initialize keystore with tomcat–vault.

```
Enter directory to store encrypted
files:/Users/weli/projs/apache-tomcat-8.0.39/conf
```

We have assigned a directory to store tomcat–vault encrypted data file. The data file is used to store username and password information, and it’ s encrypted by the key pair in the key store.

We will configure tomcat–vault to use vault.keystore, and we will also configure tomcat–vault to use the key pair my_vault in vault.keystore for encryption in following steps.

Let go on seeing the configuration process:

```
Enter Keystore URL:/Users/weli/projs/apache-tomcat-8.0.39/conf/vault.keystore
Enter Keystore password: my_password123
Enter Keystore password again: my_password123
Values match
```

Here we enter the keystore password so tomcat–vault can access it and use the key pair inside for data encryption.

Let’ s see the next step:

```
Enter 8 character salt:1234abcd
Enter iteration count as a number (Eg: 44):120
Enter Keystore Alias:my_vault
Initializing Vault
```

Here we need to enter ‘salt’ and ‘iteration count’ as we like, these two attributes are used by tomcat−vault to encrypt its data file.

After all above parameters are set, tomcat−vault will finish the initialization process and output the settings:

```
KEYSTORE_URL=/Users/weli/projs/apache-tomcat-8.0.39/conf/vault.keystore
KEYSTORE_PASSWORD=MASK-3CuP21KMHn7G6iH/A3YpM/
KEYSTORE_ALIAS=my_vault
SALT=1234abcd
ITERATION_COUNT=120
ENC_FILE_DIR=/Users/weli/projs/apache-tomcat-8.0.39/conf/
```

We need to store above config into a file. In this article, I put above config into a file named vault.properties and put it into conf directory of tomcat:

```
tb13:conf weli$ pwd
/Users/weli/projs/apache-tomcat-8.0.39/conf

tb13:conf weli$ ls vault.properties
vault.properties

tb13:conf weli$ cat vault.properties
KEYSTORE_URL=/Users/weli/projs/apache-tomcat-8.0.39/conf/vault.keystore
KEYSTORE_PASSWORD=MASK-3CuP21KMHn7G6iH/A3YpM/
KEYSTORE_ALIAS=my_vault
SALT=1234abcd
ITERATION_COUNT=120
ENC_FILE_DIR=/Users/weli/projs/apache-tomcat-8.0.39/conf/
```

From the above ENC_FILE_DIR setting, we can see the place where tomcat−vault put its data file in, and we can verify it:

```
tb13:conf weli$ pwd
/Users/weli/projs/apache-tomcat-8.0.39/conf
tb13:conf weli$ ls VAULT.dat
VAULT.dat
```

As the command output shown above, we can see the default name of the data file used by tomcat−vault is VAULT.dat.

Till now, we have three files generated into conf directory:

```
tb13:conf weli$ find . | grep -i vault
./VAULT.dat
./vault.keystore
./vault.properties
```

In production environment, you should put above files into a safer place.

Now we have finished initializing tomcat vault, the next step is to configure the tomcat to use the vault.

Configuring Tomcat To Use Vault

As we have prepared tomcat−vault for usage, now we need to configure Tomcat.