Java - encrypt / decrypt user name and password from a configuration file

Asked 16 years ago Modified 10 years, 5 months ago Viewed 84k times

14

We are busy developing a Java web service for a client. There are two possible choices:

- Store the encrypted user name / password on the web service client. Read from a config. file on the client side, decrypt and send.
- Store the encrypted user name / password on the web server. Read from a config. file on the web server, decrypt and use in the web service.

The user name / password is used by the web service to access a third-party application.

The client already has classes that provide this functionality but this approach involves sending the user name / password in the clear (albeit within the intranet). They would prefer storing the info. within the web service but don't really want to pay for something they already have. (Security is not a big consideration because it's only within their intranet).

So we need something quick and easy in Java.

Any recommendations?

The server is Tomkat 5.5. The web service is Axis2.

- What encrypt / decrypt package should we use?
- What about a key store?
- What configuration mechanism should we use?
- Will this be easy to deploy?

java encryption configuration-files

Share

Improve this question

Follow

asked Dec 3, 2008 at 22:39

rbrayb

46.7k ● 34 ● 118 ● 179

2 Answers

Sorted by: Highest score (default)

26

As I understand anyhow in order to call 3rd party web service you pass password as plain text and no security certificates are involved.

Then I would say the easiest approach would be to store password in encrypted format (via java encryption mechanism) when the encryption/decryption key is just hard coded in the code.

I would definitely store it on the server side (file system or db) rather then distribute and maintain it on the multiple

clients.

Here is how that could work with "DES" encryption:

```
// only the first 8 Bytes of the constructor argument
// as material for generating the keySpec
DESKeySpec keySpec = new DESKeySpec("YourSecr".getByte
SecretKeyFactory keyFactory = SecretKeyFactory.getInst
SecretKey key = keyFactory.generateSecret(keySpec);
sun.misc.BASE64Encoder base64encoder = new BASE64Encod
sun.misc.BASE64Decoder base64decoder = new BASE64Decod
.........

// ENCODE plainTextPassword String
byte[] cleartext = plainTextPassword.getBytes("UTF8");

Cipher cipher = Cipher.getInstance("DES"); // cipher i
cipher.init(Cipher.ENCRYPT_MODE, key);
String encrypedPwd = base64encoder.encode(cipher.doFin
// now you can store it
......

// DECODE encryptedPwd String
byte[] encrypedPwdBytes = base64decoder.decodeBuffer(e

Cipher cipher = Cipher.getInstance("DES");// cipher is
cipher.init(Cipher.DECRYPT_MODE, key);
byte[] plainTextPwdBytes = (cipher.doFinal(encrypedPwd
```

Share Improve this answer

Follow

edited Jun 23, 2011 at 20:39

Community Bot

1 • 1

answered Dec 4, 2008 at 7:09

Gennady Shumakher

5,736 • 12 • 42 • 46

Where does "Your secret Key phrase" come from? If that source is secure, why not use it to hold the actual password? – erickson Dec 4, 2008 at 17:09

As I mentioned the key phrased is hardcoded in java class, alternately it could be stored in data base. Using password itself would require to recompile the code every time the password is changed. – Gennady Shumakher Dec 4, 2008 at 17:27

3 anyone who is capable of getting access to the file can also get access to your code and decompile it (or simply use it) to obtain the key. Microsoft tried something similar to protect the SAM and it was almost immediately broken. – frankodwyer Dec 17, 2008 at 1:20

5 @frankodwyer you are right, but my answer(as well as the question itself) didn't pretend to provide fully secured solution,but rather to give some level of security. So to me it is more a question between @ericson approach: "do nothing" and other one: "there is a value to do something even minimal". – Gennady Shumakher Dec 17, 2008 at 9:36

There is a problem! If you change "Your secret Key phrase" to "Your secret Key phraseeek434jlh72eee", you'll still decrypt correctly. Wrong. – user400851 Apr 15, 2011 at 7:54

18

Being on the intranet certainly does not justify dismissing security. Most damage done to information is by insiders. Look at the value of what's being protected, and give due consideration to security.

It sounds like there's a third-party application, for which you have one set of credentials, and some clients that


effectively share this identity when using the third-party application. If that's the case, I recommend the following approach.

Don't distribute the third-party password beyond your web server.

The safest way to do this is to provide it to the web application interactively. This could be ServletContextListener that prompts for the password as the application starts, or a page in the application so that a admin can enter it through a form. The password is stored in the ServletContext and used to authenticate requests to the third-party service.

A step down in safety is to store the password on the server's file system so that it's readable only by the user running the server. This relies on the server's file system permissions for protection.

Trying to store an encrypted form of the password, on the client or the server, is just taking one step backward. You fall into an infinite regress when trying to protect a secret with another secret.

In addition, the clients should authenticate themselves to the server. If the client is interactive, have the users enter a password. The server can then decide if that user is authorized to access the third-party service. If the client is not interactive, the next best security is to protect the client's password using file system permissions.

To protect the clients' credentials, the channel between the client and your web server should be protected with SSL. Here, operating on an intranet is advantageous, because you can use a self-signed certificate on the server.

If you do store passwords in a file, put them in a file by themselves; it makes the need to manage permissions carefully more conspicuous, and minimizes the need for many users to be editing that file and thus seeing the password.

answered Dec 3, 2008 at 23:09

erickson

269k ● 59 ● 401 ● 497