Encryption

introduction

The Program Managment development team determined that in order to secure sensitive passwords or other values in the Program Management configuration properties, the most pragmatic way to go about it is to simply encrypt the data at rest in the database.

Assumptions

- It is assumed that the correct user/role permissions will be applied so that only authenticated users with the appropriate authorization will have access to the REST endpoints (and associated User Interface).
- It is also assumed that all requests are forced into an SSL session so that the data will be transmitted within an encrypted HTTP session.
- The algorithm used must be reversible as we need to be able to pass passwords in plain text to other integrating systems such as MongoDB or an SFTP site.

Design

In order to implement the encryption, the Jasypt library http://www.jasypt.org/ is used. Jasypt is a wrapper around the standard JCE built into the JDK. Jaspyt also has the ability to plug in different encryption providers other than the JCE if that is warranted. In this solution, the JCE is used.

The encryption algorithm used is generally known as Password Based Encryption (PBE) and is an implementation of PKCS #5: Password-Based Cryptography Standard. The overview of this standard can be found at RSA's website http://www.rsa.com/rsalabs/pkcs/files/h11302-wp-pkcs5v2-1-password-based-cryptography-standard.pdf

Algorithm

There are a number of different algorithms supported for PBE. The algorithm used in this solution is called PBEWithMD5AndTripleDES. A secret password is combined with a random salt value to generate a key for the encryption. The MD5 algorithm is used to create the key. The hashing algorithm is applied 1000 times (this is a configurable value). Triple DES is then used to encrypt the text. The salt value is appended to the encrypted value so that it can be used to decrypt the cipher text. This seems somewhat counter-intuitive, but it's actually quite secure. It is a very expensive operation to brute-force attack to decrypt the cipher text even though the salt is known.

Password

It is very important to keep the secret password secret. In this solution it is kept as a property in a property file. This file must be kept secure. If the password is lost or changed, any previously encrypted values will be irretrievable. Though there is currently no requirement for it, we will recommend adding a story to the backlog to create a process to re-encrypt encrypted fields in the case that a secret password change needs to be made.

References

- Along with the document on the RSA website about the PBE/PKCS standard, we found the following documents useful:
  http://www.javamex.com/tutorials/cryptography/pbe_salt.shtml - A good overview of how the salt and key generation works. It does mention that the algorithm used is probably the best of the built in algorithms, but if you can use something stronger, to do so. I believe that if something stronger is warranted, we would need to look into using a different algorithm provider.
- http://www.jasypt.org/howtoencryptuserpasswords.html - A general overview of password encryption from the Jasypt website.