# Secured Passwords

## Securing Passwords in settings.xml (MNG-553)

### Goal

Provide a way for securing passwords in settings.xml, instead of just storing it in plain text.

### Design

1. Password Obfuscation
    a. Provide a plugin which a user can use to obfuscate their passwords and use this generated secured password in their settings.xml file.
    b. Implement reading of obfuscated passwords in Maven (plexus? or maven-core?)
        - add a flag/parameter to enable the use of obfuscated passwords in Maven during the build. Add an <obfuscation> parameter in the settings.xml, as suggested by Benjamin (see comment below). In this case, Maven would check first the value of this parameter and perform un-obfuscation to the password if specified.
        - or, use a keyword prepended to the password to tell Maven that the password is obfuscated (like what Jetty does). For example, <password>OBF:securedPassword</password> with 'OBF:' as the keyword.
    c. Prompt for a password if none is found in the settings.xml. Use a parameter like the --non-interactive flag of the maven-release-plugin in order to disable this. (Already handled by Wagon as pointed out by Brett)
2. Password Encryption
   This can be implemented in two ways:
    a. Using a keystore for the server passwords
        i. Interactive:
            - Use plexus-password-store (http://svn.codehaus.org/plexus/archive/plexus-sandbox/trunk/plexus-components/plexus-password-store) to create the key store where the server credentials would be stored. Maven can also use this component for accessing the keystore as specified in the next point.
            - In Maven:
                - Add a parameter to tell Maven the location of the keystore. (Ex. -DkeystoreLocation=/path/to/keystore)
                - If the keystore location parameter is specified, Maven would disregard the <password> set in the settings.xml file (if there is any) and prompt for the master password of the keystore. The server credentials will be retrieved from the keystore if the entered master password is correct. Use the retrieved credentials to access the secured server.
                - Otherwise, use the <password> (if there is any) set in the settings.xml file. (Take into account password obfuscation here if it will be implemented.)
        ii. Non-interactive:
            - Same implementation as 'Interactive', except that the master password for the keystore can be set via a command-line parameter instead of Maven prompting for it. The drawback here is that the master password would be in plain text.
    b. Using password-based encryption on the password in <settings.xml> (see Oleg's comment below)

### Limitation(s)

Password Encryption:

1. <password> in settings.xml would be disregarded if the server credentials would be retrieved from the key store.

## References

Securing Passwords in Jetty