etiennestuder / **gradle-credentials-plugin**

Gradle plugin to store and access encrypted credentials for use in Gradle builds.

146 stars 30 forks

Star Notifications

Code Issues 1 Pull requests 1 Actions Projects Wiki Security Insights

master Go to file

etiennestuder ... ✓ 18 days ago

View code

Build Gradle project passing

gradle-credentials-plugin

> The work on this software project is in no way associated with my employer nor with the role I'm having at my employer. Any requests for changes will be decided upon exclusively by myself based on my personal preferences. I maintain this project as much or as little as my spare time permits.

Overview

Gradle plugin that allows to store and access encrypted credentials using password-based encryption (PBE).

The credentials plugin is hosted at Bintray's JCenter.

Build scan

Recent build scan: https://scans.gradle.com/s/vzlk2e4dnfzge

Find out more about build scans for Gradle and Maven at https://scans.gradle.com.

Goals

One typical use case of the 'gradle.properties' file in the Gradle user home directory is to store credentials, and to reference them from Gradle builds as project properties. This is a very convenient functionality at the cost that, by default, these properties are stored in plain text. It happens quickly that such credentials are exposed accidentally while giving a Gradle presentation or while pair-programming with a colleague.

The credentials plugin provides a parallel functionality to the 'gradle.properties' file to store and access credentials in an encrypted format through a 'gradle.encrypted.properties' files, thereby avoiding that credentials are ever stored in plain text.

Functionality

The following functionality is provided by the credentials plugin:

- Store encrypted credentials
- Delete encrypted credentials
- Access encrypted credentials from a Gradle build

Design

All access and storage of credentials goes through password-based encryption. The passphrase can either be specified as a project property from the command line, or a default passphrase is used. The JDK encryption algorithm applied is *AES* using a key that is generated using *PBKDF2WithHmacSHA1* from an 8-byte salt, an iteration count of 65536, and a key length of 128 (longer keys require local installation of the JRE Security Extension).

Access to the stored credentials from within a Gradle build happens through the `credentials` project property. All read and write operations to the credentials container apply the decryption and encryption on the fly. The credentials container never holds any credentials in their decrypted form.

Please note that the author of this plugin is by far not a security expert. It is also not the primary goal of this plugin to provide high-security encryption, but rather to provide a convenient way to avoid having to store credentials in plain text.

Configuration

Apply credentials plugin

Project-application

Apply the `nu.studer.credentials` plugin to your Gradle project.

```
plugins {
  id 'nu.studer.credentials' version '2.1'
}
```

Please refer to the Gradle DSL PluginDependenciesSpec to understand the behavior and limitations when using the new syntax to declare plugin dependencies.

Settings-application

Apply the `nu.studer.credentials` plugin to your Gradle settings file.

```
buildscript {
    repositories {
        gradlePluginPortal()
    }
    dependencies {
        classpath 'nu.studer:gradle-credentials-plugin:2.1'
    }
}

apply plugin: 'nu.studer.credentials'
```

Invoke credentials tasks

Store encrypted credentials

You can store new credentials or update existing credentials through the `addCredentials` task. Pass along the credentials key and value through the task options `--key` and `--value` . The credentials are stored in the *GRADLE_USER_HOME/gradle.encrypted.properties*.

```
gradle addCredentials --key someKey --value someValue
```

Optionally, pass along a custom passphrase through the `credentialsPassphrase` project property. The credentials are stored in the passphrase-specific *GRADLE_USER_HOME/gradle.MD5HASH.encrypted.properties* where the *MD5HASH* is calculated from the specified passphrase.

```
gradle addCredentials --key someKey --value someValue -
PcredentialsPassphrase=mySecretPassPhrase
```

Optionally, pass along a custom directory location of the credentials file through the `credentialsLocation` project property.

```
gradle addCredentials --key someKey --value someValue -
PcredentialsLocation=/some/directory
```

Remove encrypted credentials

You can remove existing credentials through the `removeCredentials` task. Pass along the credentials key through the `--key` project property. The credentials are removed from the *GRADLE_USER_HOME/gradle.encrypted.properties*.

```
gradle removeCredentials --key someKey
```

Optionally, pass along a custom passphrase through the `credentialsPassphrase` project property. The credentials are removed from the passphrase-specific *GRADLE_USER_HOME/gradle.MD5HASH.encrypted.properties* where the *MD5HASH* is calculated from the specified passphrase.

```
gradle removeCredentials --key someKey -PcredentialsPassphrase=mySecretPassPhrase
```

Optionally, pass along a custom directory location of the credentials file through the `credentialsLocation` project property.

```
gradle addCredentials --key someKey --value someValue -
PcredentialsLocation=/some/directory
```

Access credentials in build

Get credentials from within a build

Get the desired credentials from the `credentials` container, available on the project instance. The credentials are decrypted as they are accessed.

```
String accountPassword = credentials.someAccountName
```

If no explicit passphrase is passed when starting the build, the `credentials` container is initialized with all credentials persisted in the *GRADLE_USER_HOME/gradle.encrypted.properties*.

If a custom passphrase is passed through the `credentialsPassphrase` project property when starting the build, the `credentials` container is initialized with all credentials persisted in the passphrase-specific *GRADLE_USER_HOME/gradle.MD5HASH.encrypted.properties* where the *MD5HASH* is calculated from the specified passphrase.

If a custom directory location is passed through the `credentialsLocation` project property when starting the build, the credentials file will be seeked in that directory.

Add credentials ad-hoc from within a build

Set the desired credentials on the `credentials` container, available on the project instance. The credentials are encrypted as they are assigned.

README.md

Credentials added ad-hoc during the build are not persisted on the file system.

Example

Project-application

The build script of the credentials plugin makes use of itself and serves as a real-world example. You can also find a self-contained example build script here.

Settings-application

The credentials plugin can also be applied to a Gradle settings file. You can find a self-contained example build script here.

Feedback and Contributions

Both feedback and contributions are very welcome.

Acknowledgements

- Myllyenko (pr)
- aingram (pr)

License

Releases 10

Support for settings.gradle Latest
on 27 Dec 2019

+ 9 releases

Contributors 3

etiennestuder Etienne Studer

runningcode Nelson Osacky

aingram Aaron Ingram

Languages

● **Java** 66.2% ● **Groovy** 33.8%