rules.sonarsource.com

# XML static code analysis: Hard-coded credentials are security-sensitive

3-4 minutes

---

Because it is easy to extract strings from an application source code or binary, credentials should not be hard-coded. This is particularly true for applications that are distributed or that are open-source.

In the past, it has led to the following vulnerabilities:

- CVE-2019-13466
- CVE-2018-15389

Credentials should be stored outside of the code in a configuration file, a database, or a management service for secrets.

This rule flags instances of hard-coded credentials used in database and LDAP connections. It looks for hard-coded credentials in connection strings, and for variable names that match any of the patterns from the provided list.

It's recommended to customize the configuration of this rule with additional credential words such as "oauthToken", "secret", …

## Ask Yourself Whether

- Credentials allows access to a sensitive component like a database, a file storage, an API or a service.
- Credentials are used in production environments.
- Application re-distribution is required before updating the credentials.

There is a risk if you answered yes to any of those questions.

## Recommended Secure Coding Practices

- Store the credentials in a configuration file that is not pushed to the code repository.
- Store the credentials in a database.
- Use your cloud provider's service for managing secrets.
- If a password has been disclosed through the source code: change it.

## Sensitive Code Example

Spring-social-twitter secrets can be stored inside a xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="connectionFactoryLocator"
class="org.springframework.social.connect.support.ConnectionFactoryRegistry
      <property name="connectionFactories">
          <list>
              <bean
class="org.springframework.social.twitter.connect.TwitterConnectionFactory">
                  <constructor-arg value="username" />
                  <constructor-arg value="very-secret-password" />   <!--
Sensitive -->
              </bean>
          </list>
      </property>
  </bean>
</beans>
```

## Compliant Solution

In spring social twitter, retrieve secrets from environment variables:

```
@Configuration
public class SocialConfig implements SocialConfigurer {

    @Override
    public void
addConnectionFactories(ConnectionFactoryConfigurer cfConfig,
Environment env) {
        cfConfig.addConnectionFactory(new
TwitterConnectionFactory(
            env.getProperty("twitter.consumerKey"),
            env.getProperty("twitter.consumerSecret")));  <!-- Compliant
-->
    }
}
```

## See

- OWASP Top 10 2021 Category A7 - Identification and Authentication Failures
- OWASP Top 10 2017 Category A2 - Broken Authentication
- MITRE, CWE-798 - Use of Hard-coded Credentials
- MITRE, CWE-259 - Use of Hard-coded Password
- SANS Top 25 - Porous Defenses
- Derived from FindSecBugs rule Hard Coded Password