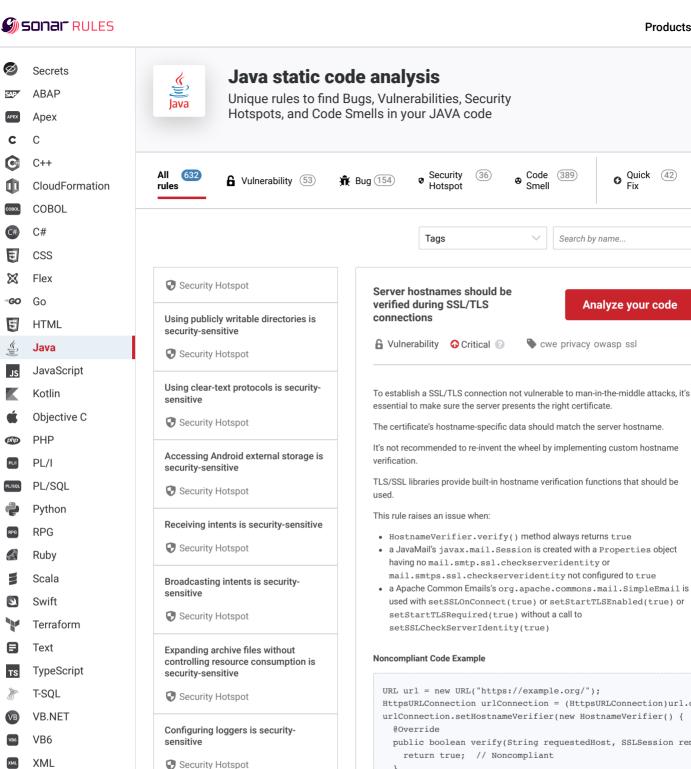


Products ✓

O Quick 42 Fix



Using weak hashing algorithms is

Using unsafe Jackson deserialization

configuration is security-sensitive

Setting JavaBean properties is

Disabling CSRF protections is

security-sensitive

Security Hotspot

Security Hotspot

security-sensitive Security Hotspot

security-sensitive

Security Hotspot

```
HttpsURLConnection urlConnection = (HttpsURLConnection)url.o
urlConnection.setHostnameVerifier(new HostnameVerifier() {
  public boolean verify(String requestedHost, SSLSession rem
});
InputStream in = urlConnection.getInputStream();
```

SimpleEmail example:

```
Email email = new SimpleEmail();
email.setSmtpPort(465);
email.setAuthenticator(new DefaultAuthenticator(username, pa
email.setSSLOnConnect(true); // Noncompliant; setSSLCheckSer
email.send();
```

JavaMail's example:

```
Properties props = new Properties();
props.put("mail.smtp.host", "smtp.gmail.com");
props.put("mail.smtp.socketFactory.port", "465");
props.put("mail.smtp.socketFactory.class", "javax.net.ssl.SS
props.put("mail.smtp.auth", "true");
props.put("mail.smtp.port", "465");
```

Using non-standard cryptographic algorithms is security-sensitive

Security Hotspot

Using pseudorandom number generators (PRNGs) is securitysensitive

Security Hotspot

Mocking all non-private methods of a class should be avoided

Code Smell

Empty lines should not be tested with regex MULTILINE flag

```
Session session = Session.getDefaultInstance(props, new java
  protected PasswordAuthentication getPasswordAuthentication
    return new PasswordAuthentication("username@gmail.com",
});
```

Compliant Solution

```
URL url = new URL("https://example.org/");
HttpsURLConnection urlConnection = (HttpsURLConnection)url.o
// Compliant; Use the default HostnameVerifier
InputStream in = urlConnection.getInputStream();
```

SimpleEmail example:

```
Email email = new SimpleEmail();
email.setSmtpPort(465);
email.setAuthenticator(new DefaultAuthenticator(username, pa
email.setSSLOnConnect(true);
email.setSSLCheckServerIdentity(true); // Compliant
email.send();
```

JavaMail's example:

```
Properties props = new Properties();
props.put("mail.smtp.host", "smtp.gmail.com");
props.put("mail.smtp.socketFactory.port", "465");
props.put("mail.smtp.socketFactory.class", "javax.net.ssl.SS
props.put("mail.smtp.auth", "true");
props.put("mail.smtp.port", "465");
props.put("mail.smtp.ssl.checkserveridentity", true); // Com
Session session = Session.getDefaultInstance(props, new java
  protected PasswordAuthentication getPasswordAuthentication
    return new PasswordAuthentication("username@gmail.com",
});
```

See

- OWASP Top 10 2021 Category A2 Cryptographic Failures
- OWASP Top 10 2021 Category A5 Security Misconfiguration
- OWASP Top 10 2021 Category A7 Identification and Authentication Failures
- OWASP Top 10 2017 Category A6 Security Misconfiguration
- MITRE, CWE-295 Improper Certificate Validation
- Derived from FindSecBugs rule WEAK_HOSTNAME_VERIFIER

Available In:

sonarlint ⊕ | sonarcloud ♦ | sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved. Privacy Policy