




 Secrets


 ABAP


 Apex


 C


 C++


 CloudFormation


 COBOL


 C#


 CSS


 Flex


 Go


 HTML


 **Java**


 JavaScript


 Kotlin


 Objective C


 PHP


 PL/I


 PL/SQL


 Python


 RPG


 Ruby


 Scala


 Swift


 Terraform


 Text


 TypeScript

 T-SQL

 VB.NET

 VB6

 XML



# Java static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVA code

All rules632

Vulnerability53

Bug154

Security Hotspot36

Code Smell389

Quick Fix42

Tags ▾

Search by name... 🔍

Security Hotspot

Using publicly writable directories is security-sensitive

Security Hotspot

Using clear-text protocols is security-sensitive

Security Hotspot

Accessing Android external storage is security-sensitive

Security Hotspot

Receiving intents is security-sensitive

Security Hotspot

Broadcasting intents is security-sensitive

Security Hotspot

Expanding archive files without controlling resource consumption is security-sensitive

Security Hotspot

Configuring loggers is security-sensitive

Security Hotspot

Using weak hashing algorithms is security-sensitive

Security Hotspot

Using unsafe Jackson deserialization configuration is security-sensitive

Security Hotspot

Setting JavaBean properties is security-sensitive

Security Hotspot

Disabling CSRF protections is security-sensitive

Security Hotspot

Server hostnames should be verified during SSL/TLS connections

Analyze your code

VulnerabilityCritical🔍cwe privacy owasp ssl

To establish a SSL/TLS connection not vulnerable to man-in-the-middle attacks, it's essential to make sure the server presents the right certificate.

The certificate's hostname-specific data should match the server hostname.

It's not recommended to re-invent the wheel by implementing custom hostname verification.

TLS/SSL libraries provide built-in hostname verification functions that should be used.

This rule raises an issue when:

- HostnameVerifier.verify() method always returns true
- a JavaMail's javax.mail.Session is created with a Properties object having no mail.smtp.ssl.checkserveridentity or mail.smtps.ssl.checkserveridentity not configured to true
- a Apache Commons Emails's org.apache.commons.mail.SimpleEmail is used with setSSLonConnect(true) or setStartTLSEnabled(true) or setStartTLSRequired(true) without a call to setSSLCheckServerIdentity(true)

Noncompliant Code Example

```
URL url = new URL("https://example.org/");
HttpsURLConnection urlConnection = (HttpsURLConnection)url.openConnection().setHostnameVerifier(new HostnameVerifier() {
    @Override
    public boolean verify(String requestedHost, SSLSession rem
        return true; // Noncompliant
    }
});
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");
```

https://rules.sonarsource.com/java/RSPEC-5527

1/2

<div>Using non-standard cryptographic algorithms is security-sensitive</div> <div> Security Hotspot</div>
<div>Using pseudorandom number generators (PRNGs) is security-sensitive</div> <div> Security Hotspot</div>
<div>Mocking all non-private methods of a class should be avoided</div> <div> Code Smell</div>
<div>Empty lines should not be tested with regex MULTILINE flag</div> <div> Code Smell</div>

```
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/");
URLConnection urlConnection = (URLConnection)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 