





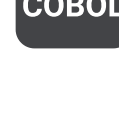

























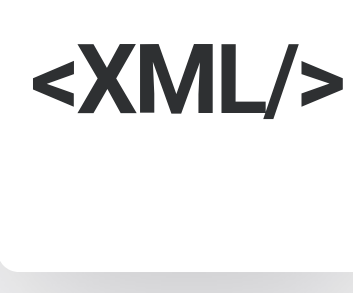


-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  PL/SQL
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  **XML**

 **<XML/>**

## XML static code analysis

Unique rules to find Bugs and Code Smells in your XML code

All rules 36


 Vulnerability 6













 Bug 5

 Security Hotspot 9

 Code Smell 16

Tags ▾

Search by name... 

Using clear-text protocols is security-sensitive	 Security Hotspot
Receiving intents is security-sensitive	 Security Hotspot
Restrict access to exported components with appropriate permissions	 Vulnerability
"DefaultMessageListenerContainer" instances should not drop messages during restarts	 Bug
"SingleConnectionFactory" instances should be set to "reconnectOnException"	 Bug
Defining a single permission for read and write access of Content Providers is security-sensitive	 Security Hotspot
Allowing application backup is security-sensitive	 Security Hotspot
Requesting dangerous Android permissions is security-sensitive	 Security Hotspot
Sections of code should not be commented out	 Code Smell
Track uses of "FIXME" tags	 Code Smell
Custom permissions should not be defined in the 'android.permission' namespace	 Vulnerability
Having a permissive Cross-Origin Resource Sharing policy is security-sensitive	 Security Hotspot

### Using clear-text protocols is security-sensitive

Analyze your code

 Security Hotspot

 Critical



 cwe owasp

Clear-text protocols such as `ftp`, `telnet` or non-secure `http` lack encryption of transported data, as well as the capability to build an authenticated connection. It means that an attacker able to sniff traffic from the network can read, modify or corrupt the transported content. These protocols are not secure as they expose applications to an extensive range of risks:

- Sensitive data exposure
- Traffic redirected to a malicious endpoint
- Malware infected software update or installer
- Execution of client side code
- Corruption of critical information

Even in the context of isolated networks like offline environments or segmented cloud environments, the insider threat exists. Thus, attacks involving communications being sniffed or tampered with can still happen.

For example, attackers could successfully compromise prior security layers by:

- Bypassing isolation mechanisms
- Compromising a component of the network
- Getting the credentials of an internal IAM account (either from a service account or an actual person)

In such cases, encrypting communications would decrease the chances of attackers to successfully leak data or steal credentials from other network components. By layering various security practices (segmentation and encryption, for example), the application will follow the *defense-in-depth* principle.

Note that using the `http` protocol is being deprecated by [major web browsers](#).

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

- [CVE-2019-6169](#)
- [CVE-2019-12327](#)
- [CVE-2019-11065](#)

#### Ask Yourself Whether

- Application data needs to be protected against falsifications or leaks when transiting over the network.
- Application data transits over a network that is considered untrusted.
- Compliance rules require the service to encrypt data in transit.
- Your application renders web pages with a relaxed mixed content policy.
- OS level protections against clear-text traffic are deactivated.

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

#### Recommended Secure Coding Practices

- Make application data transit over a secure, authenticated and encrypted protocol like TLS or SSH. Here are a few alternatives to the most common clear-text protocols:
  - Use `ssh` as an alternative to `telnet`
  - Use `sftp`, `scp` or `ftps` instead of `ftp`
  - Use `https` instead of `http`
  - Use SMTP over SSL/TLS or SMTP with STARTTLS instead of clear-text SMTP
- Enable encryption of cloud components communications whenever it's possible.
- Configure your application to block mixed content when rendering web pages.
- If available, enforce OS level deactivation of all clear-text traffic

It is recommended to secure all transport channels (even local network) as it can take a single non secure connection to compromise an entire application or system.

#### Sensitive Code Example

```
<application
  android:usesCleartextTraffic="true"> <!-- Sensitive -->
</application>
```

For versions older than Android 9 (API level 28) `android:usesCleartextTraffic` is implicitly set to `true`

```
<application> <!-- Sensitive -->
</application>
```

#### Compliant Solution

```
<application
  android:usesCleartextTraffic="false">
</application>
```

#### See

- [OWASP Top 10 2021 Category A2](#) - Cryptographic Failures
- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [Mobile AppSec Verification Standard](#) - Network Communication Requirements
- [OWASP Mobile Top 10 2016 Category M3](#) - Insecure Communication
- [MITRE, CWE-200](#) - Exposure of Sensitive Information to an Unauthorized Actor
- [MITRE, CWE-319](#) - Cleartext Transmission of Sensitive Information
- [Google, Moving towards more secure web](#)
- [Mozilla, Deprecating non secure http](#)

Available In: