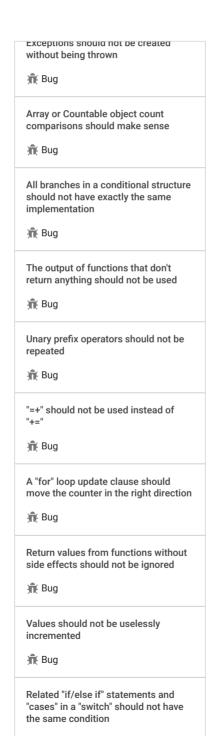




PHP static code analysis

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

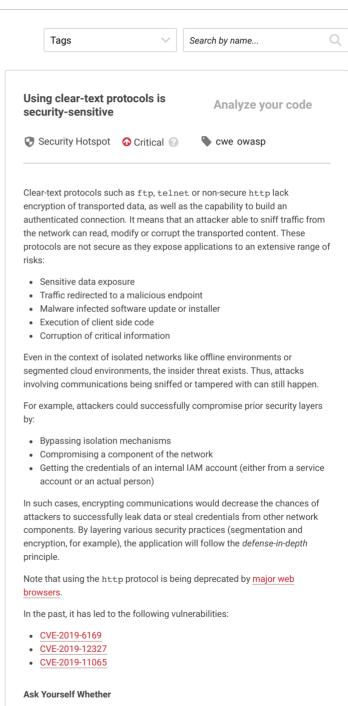




R Bug

used

Objects should not be created to be dropped immediately without being



· Application data needs to be protected against falsifications or leaks

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.

when transiting over the network.

Recommended Secure Coding Practices

™ Bug

Identical expressions should not be used on both sides of a binary operator



All code should be reachable

👬 Bug

Loops with at most one iteration should be refactored



Short-circuit logic should be used to prevent null pointer dereferences in conditionals



- 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:
 - Usessh 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 cleartext 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 deativation 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

```
$url = "http://example.com"; // Sensitive
$url = "ftp://anonymous@example.com"; // Sensitive
$url = "telnet://anonymous@example.com"; // Sensitive
$con = ftp_connect('example.com'); // Sensitive

$trans = (new Swift_SmtpTransport('XXX', 1234)); // Sen
$mailer = new PHPMailer(true); // Sensitive

define( 'FORCE_SSL_ADMIN', false); // Sensitive

define( 'FORCE_SSL_LOGIN', false); // Sensitive
```

Compliant Solution

```
$url = "https://example.com"; // Compliant
$url = "sftp://anonymous@example.com"; // Compliant
$url = "ssh://anonymous@example.com"; // Compliant
$con = ftp_ssl_connect('example.com'); // Compliant
$trans = (new Swift_SmtpTransport('smtp.example.org', 1
    ->setEncryption('tls') // Compliant
;

$mailer = new PHPMailer(true);
$mailer->SMTPSecure = 'tls'; // Compliant
define( 'FORCE_SSL_ADMIN', true); // Compliant
define( 'FORCE_SSL_LOGIN', true); // Compliant
```

Exceptions

No issue is reported for the following cases because they are not considered sensitive:

 Insecure protocol scheme followed by loopback addresses like 127.0.0.1 or localhost

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:



© 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