Test case 1

ASVS V2.1 Password Security
Unique ID: 2.1.1-1
CWE: 521
Description: Verify that user set passwords are at least 12 characters in length (after multiple spaces are combined).

Repeatable step:

1. Launch OpenEMR using URL: http://localhost:80/
2. Login using credentials
3. Click on the user profile icon on the top right corner and then in the drop down menu, select change password to redirect to that page.
4. Enter current password.
5. Enter a new password with length less than 12 characters.
6. Re-enter new password.
7. Click on save changes.

Expected results:

1. Unable to change password.
2. The error should be shown as “Password too short. Minimum characters required: 12”

Test case 2

ASVS V2.1 Password Security
Unique ID: 2.1.2-1
CWE: 521
Description: Verify that passwords of at least 64 characters are permitted, and that passwords of more than 128 characters are denied.

Repeatable step:

1. Launch OpenEMR using URL: http://localhost:80/
2. Login using credentials
3. Click on user profile icon on the top right corner and then in drop down menu, select change password to redirect to that page.
4. Enter the current password.
5. Enter a new password with a character length of atleast 64 and upto a character length 128.
6. Re-enter the new password.
7. Click on save changes.

Expected results:

1. Allows to set a 64 character password to 127 character password.
2. Denies to set a 128 character password.

Test case 3

ASVS V2.1 Password Security
Unique ID: 2.1.3-1
CWE: 521
Description: Verify that password truncation is not performed. However, consecutive multiple spaces may be replaced by a single space.

Repeatable step:

1. Launch OpenEMR using URL: http://localhost:80/
2. Login using credentials.
3. Click on the user profile icon on the top right corner and then in drop down menu, select change password to redirect to that page.
4. Enter the current password.
5. Enter a new password with a character length 70 which may also contain spaces.
6. Re-enter the new password.
7. Click on save changes.

Expected results:
1. Password will be changed without truncation.
2. Multiple consecutive spaces in password would not be replaced by single space

Test case 4

ASVS V2.1 Password Security
Unique ID: 2.1.4-1
CWE: 521
Description: Verify that any printable Unicode character, including language neutral characters such as spaces and Emojis are permitted in passwords.

Repeatable step:
1. Launch OpenEMR using URL: http://localhost:80/
2. Login using credentials.
3. Click on user profile icon on the top right corner and then in drop down menu, select change password to redirect to that page.
4. Enter the current password.
5. Enter a new password with at least one character which is printable Unicode character including emojis and spaces.
6. Re-enter the same new password.
7. Click on save changes.

Expected results:
1. Allow setting a password with this combination.

Test case 5

ASVS V2.1 Password Security
Unique ID: 2.1.5-1
CWE: 620
Description: Verify users can change their password.

Repeatable step:

1. Launch OpenEMR using URL: http://localhost:80/
2. Login using credentials
3. Click on user profile icon on the top right corner and then in drop down menu, select change password to redirect to that page.
4. Enter the current password.
5. Enter a new password.
6. Re-enter the new password.
7. Click on save changes

Expected results:

1. You able to change the password.

Test case 6

ASVS V2.1 Password Security
Unique ID: 2.1.6-1
CWE: 620
Description: Verify that password change functionality requires the user's current and new password.

Repeatable step:

1. Launch OpenEMR using URL: http://localhost:80/
2. Login using credentials.
3. Click on user profile icon on the top right corner and then in drop down menu, select change password to redirect to that page.
4. Do not enter the current password, leave it blank.
5. Enter a new password
6. Re-enter the same new password.
7. Click on save changes.

Expected results:

1. Does not allow to set new password.
2. Shows the error message that “Current password is required”

Test case 7

ASVS V2.1 Password Security
Unique ID: 2.1.8-1
CWE: 521
Description: Verify that a password strength meter is provided to help users set a stronger password.

Repeatable step:

1. Launch OpenEMR using URL: http://localhost:80/
2. Log in using credentials
3. Click on user profile icon on the top right corner and then in drop down menu, select change password to redirect to that page.
4. Enter the current password.
5. Enter a new password.
6. Re-enter the new password.

Expected results:

1. It should show password strength meter describing the strength of the new password entered.

Test case 8

ASVS V2.1 Password Security
Unique ID: 2.1.11-1
CWE: 521
Description: Verify that "paste" functionality, browser password helpers, and external password managers are permitted.

Repeatable step:

1. Launch OpenEMR using URL: http://localhost:80/ which takes us to login page
2. Enter username.
3. Try to paste password from the clipboard or password managers such as google password manager.

Expected results:

1. Password is pasted in the password field from the clipboard or password manager.

Test case 9

ASVS V2.1 Password Security
Unique ID: 2.1.12-1
CWE: 521
Description: Verify that the user can choose to either temporarily view the entire masked password, or temporarily view the last typed character of the password on platforms that do not have this as built-in functionality.

Repeatable step:

1. Launch OpenEMR using URL http://localhost:80/ that opens the login page
2. Fill up the login credentials

Expected results:

1. The user should be able to choose to either temporarily view the entire masked password, or temporarily view the last typed character of the password

Test case 10

ASVS V3.1 Fundamental Session Management Security
Unique ID: 3.1.1-1
CWE: 598
Description: Verify the application never reveals session tokens in URL parameters.

Repeatable step:

1. Launch OpenEMR using URL: http://localhost:80/
2. Log in to the website
3. After login, you will be redirected to the home page.

Expected results:

1. No token should be displayed on url.

Test case 11

ASVS V3.3 Session Termination
Unique ID: 3.3.1-1
CWE: 613
Description: Verify that logout and expiration invalidate the session token, such that the back button or a downstream relying party does not resume an authenticated session, including across relying parties

Repeatable step:

1. Launch OpenEMR using URL: http://localhost:80/
2. Log in using credentials
3. Click on user profile icon on the top right corner and then in drop down menu, select Logout. It redirects you to login page.
4. Now hit back button of browser.

Expected results:

1. It should not resume the previously authenticated session and not able to access the authorized content.

Test case 12

ASVS V3.3 Session Termination
Unique ID: 3.3.3-1
CWE: 613
Description: Verify that the application gives the option to terminate all other active sessions after a successful password change (including change via password reset/recovery), and that this is effective across the application, federated login (if present), and any relying parties.

Repeatable step:

1. Launch OpenEMR using URL: http://localhost:80/
2. Log in using credentials
3. Click on user profile icon on the top right corner and then in drop down menu, select change password to redirect to that page.
4. Enter the current password.
5. Enter a new password.
6. Re-enter the new password.
7. Click on save changes.

Expected results:

1. After changing password It should show the option to terminate all other active sessions after a successful password change.

Test case 13

ASVS V3.3 Session Termination
Unique ID: 3.3.4-1
CWE: 613
Description: Verify that users are able to view and (having re-entered login credentials) log out of any or all currently active sessions and devices.

Repeatable step:
1. Launch OpenEMR using URL: http://localhost:80/
2. Login using credentials.
3. Click on the user profile icon on the top right corner to look for active sessions.

Expected results:
1. There should be an option to view and (having re-entered login credentials) log out of any or all currently active sessions and devices.

Test case 14

ASVS V12.1 File Upload
Unique ID: 12.1.1-1
CWE: 400
Description: Verify that the application will not accept large files that could fill up storage or cause a denial of service

Repeatable step:
1. Launch OpenEMR using URL: http://localhost:80/
2. Log in using credentials
3. Click on patients on navigation bar.
4. In dropdown menu select New/Search
5. Now search existing patients or create new patients.
6. Now in Medical Record Dashboard of given patients select documents in navbar.
7. Select on lab reports
8. Now select file with size more than 300MB to upload.
9. Click on upload

Expected results:
1. It should not allow to upload the file.
2. Error should be displayed “Cannot accept this large file”.

Test case 15

ASVS V5.3 Output Encoding and Injection Prevention
Unique ID: 5.3.4-1
CWE: 89
Description: Verify that data selection or database queries (e.g. SQL, HQL, ORM, NoSQL) use parameterized queries, ORMs, entity frameworks, or are otherwise protected from database injection attacks

Repeatable step:

10. Launch OpenEMR using URL: http://localhost:80/
11. Provide Username as 1" or "1"="1_ and Password_ as_ 1" or "1" = "1
12. Click Login

Expected results:

3. Not able to login the user.
4. Error should be displayed “Invaild username/password”.

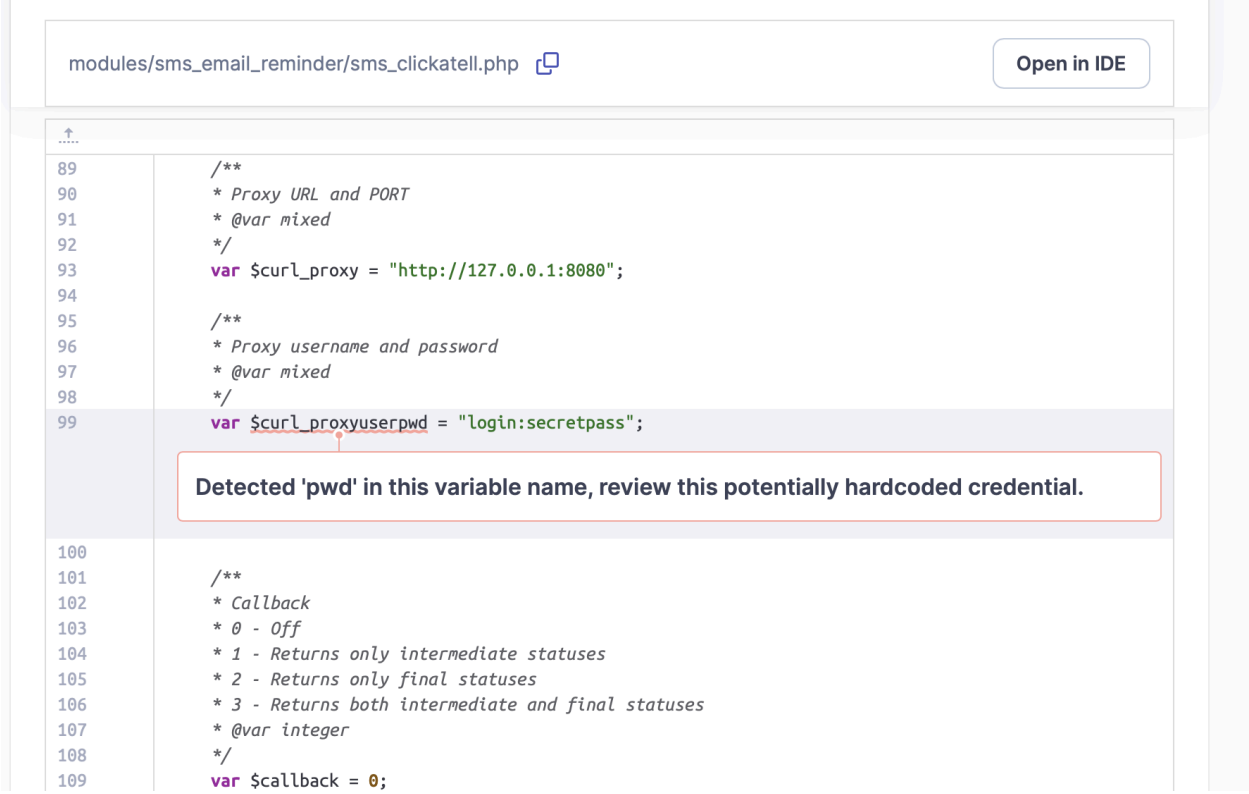
First Part Final Timing :

Number of hours spent : 15
Number of True Positive Vulnerabilities: 7
Number of True Positive Vulnerabilities per hour : 7/15

Project Part 2 : Static application security testing:

True Positives :

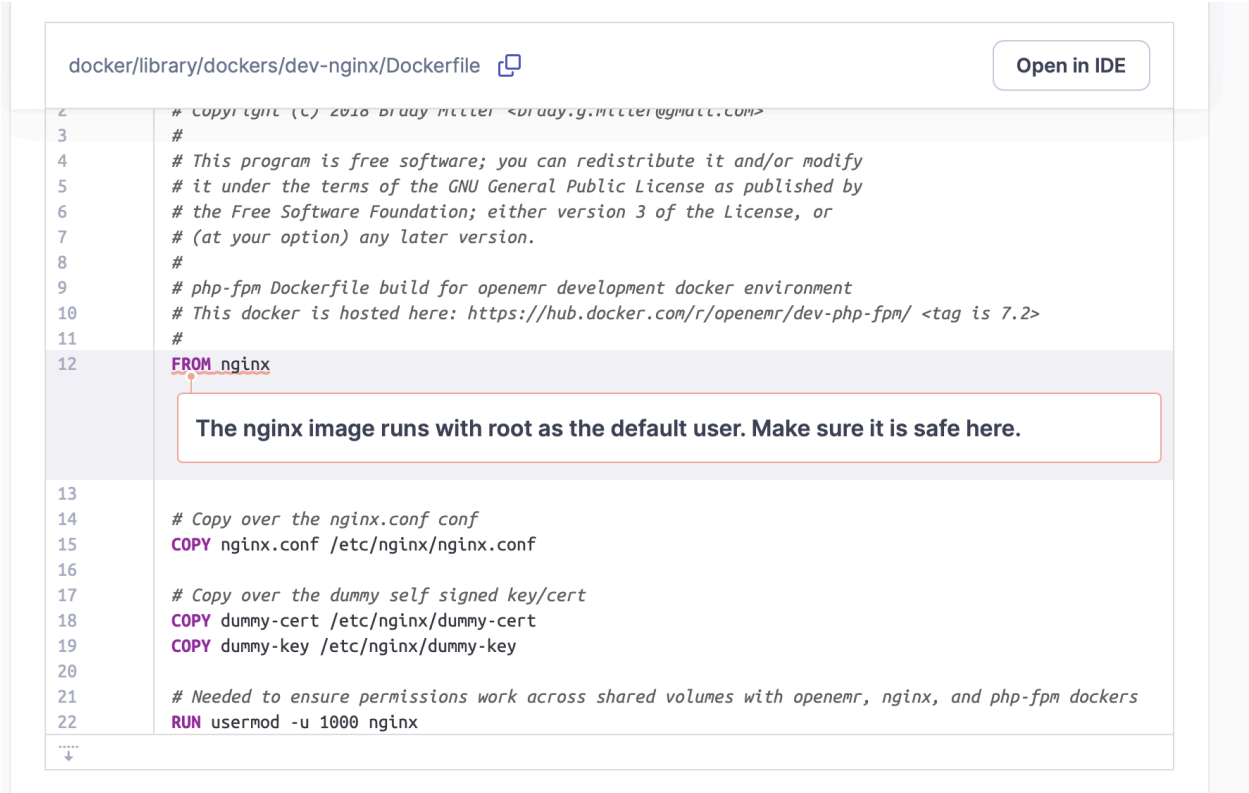
1. **Hardcoded credential in pwd variable**
   Cross Reference: modules/sms_email_reminder/sms_clickatell.php
   CWE: 798, 259
   ASVS: 6.4.1



Solution : Mitigate security risks by securely storing sensitive credentials in a dedicated file or utilizing a password vault. Strengthen overall security posture by enforcing stringent access controls, conducting regular audits, and maintaining up-to-date credentials. Additionally, consider implementing multi-factor authentication for an added layer of protection against unauthorized access and do not hardcode the passwords.

2. Default user is set to root for libraries

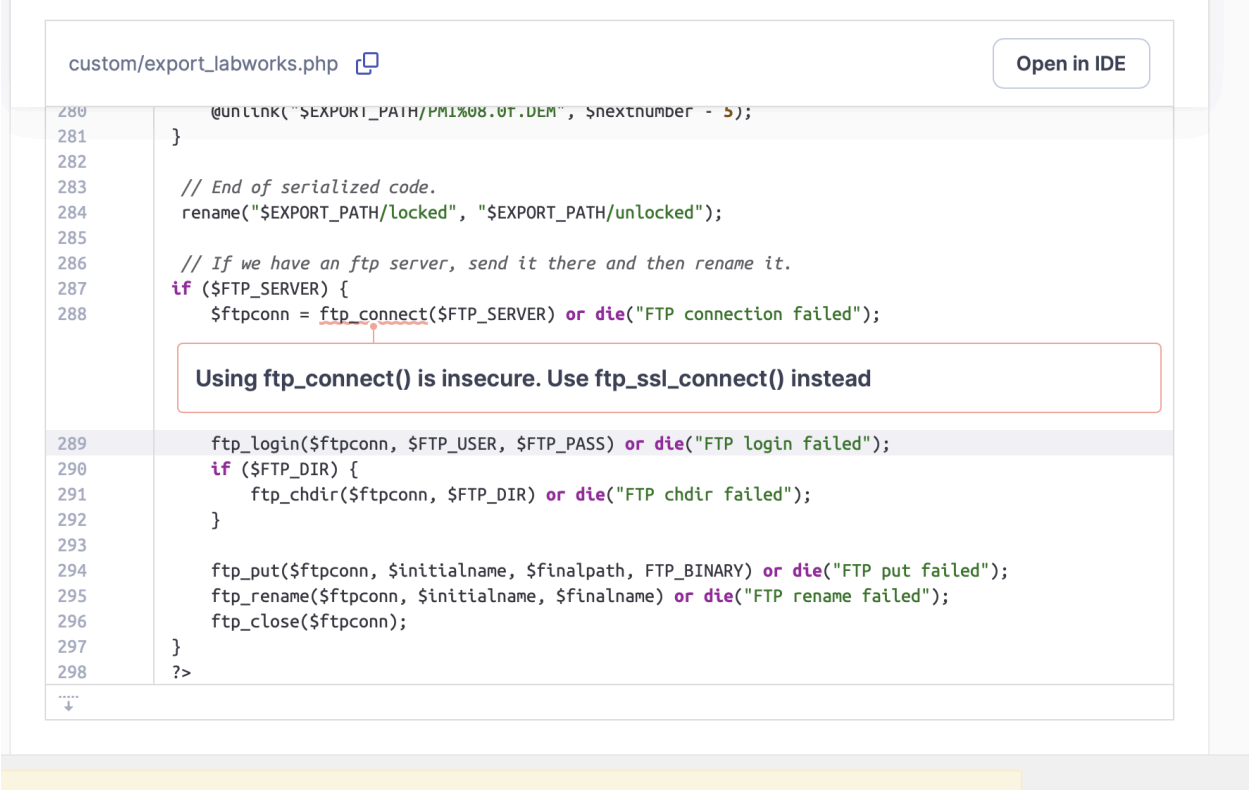
Cross Reference: docker/library/dockers/dev-nginx/Dockerfile
CWE: 250.
ASVS: 2.10.2



Solution: Enhance security measures by establishing a distinct non-root user and associating it with the USER parameter. This practice helps minimize potential vulnerabilities and adheres to the principle of least privilege. Consider implementing regular reviews and updates to user permissions to fortify the overall security posture of the system

3. Use of ftp_connect to transmit data instead of encrypted of ftp_ssl_connect

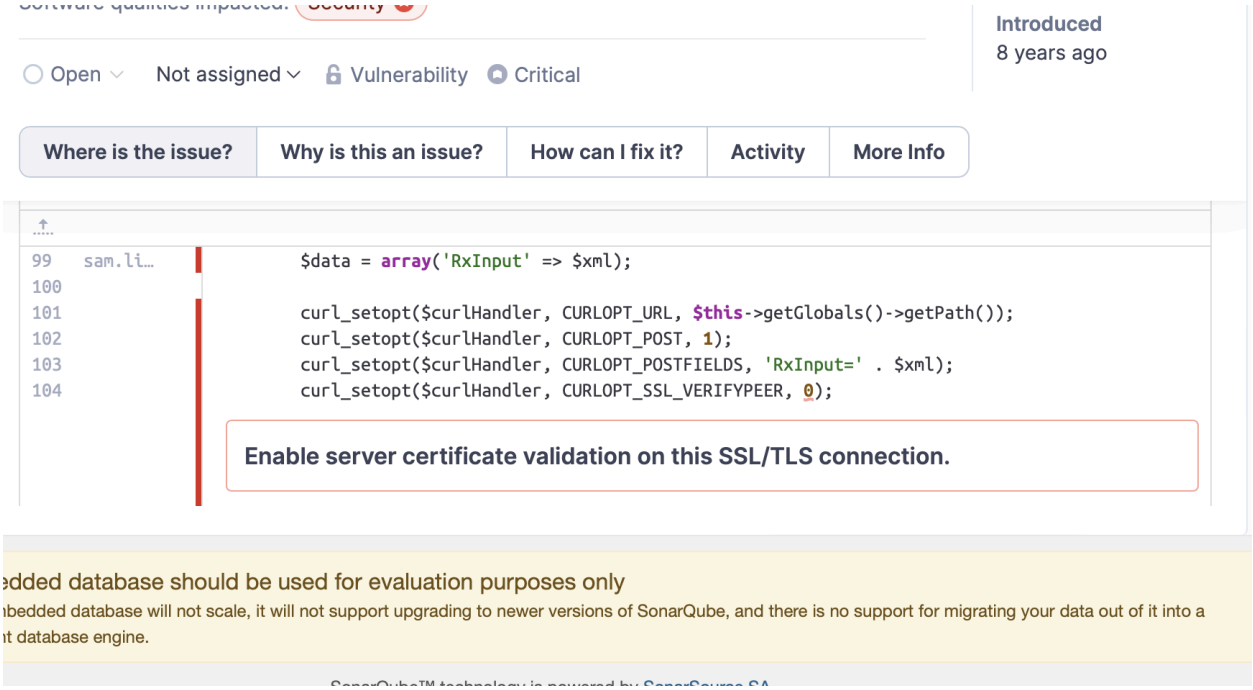
Cross Reference: custom/export_labworks.php
CWE: 311 - Missing Encryption of Sensitive Data
ASVS: 9.2.1



Solution: Safeguard sensitive data transmission by opting for ftp_ssl_connect() instead of ftp_connect(). This proactive measure enhances security by encrypting the communication channel. Regularly review and update protocols for handling sensitive data to uphold a robust security framework.

4. Improper server certificate validation on SSL/TLS connection

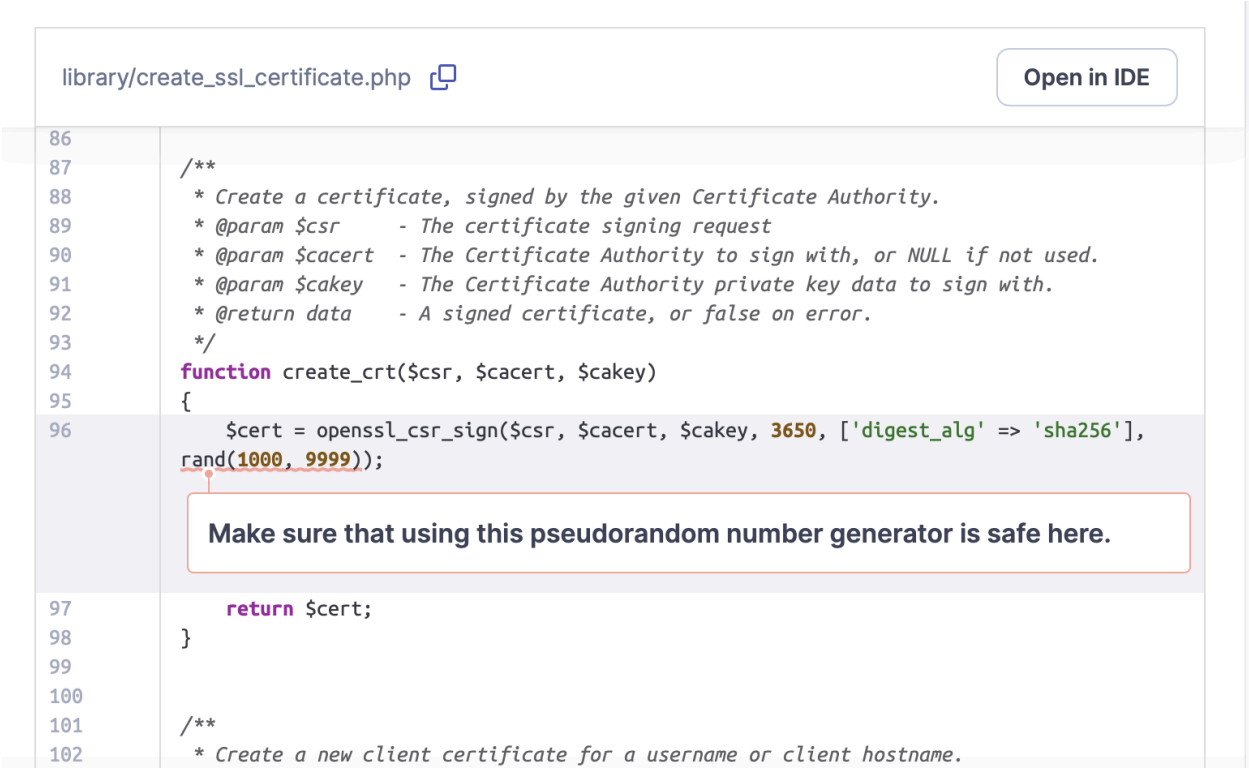
Cross Reference: interface/eRxXMLBuilder.php
CWE: 295
ASVS: 1.9.2



Solution: Strengthen security practices by enabling certificate verification through the configuration of `CURLOPT_SSL_VERIFYPEER`, setting it to 1. This precautionary step enhances the authenticity of SSL/TLS connections. Regularly review and update security configurations to ensure the continuous protection of sensitive data during communication.

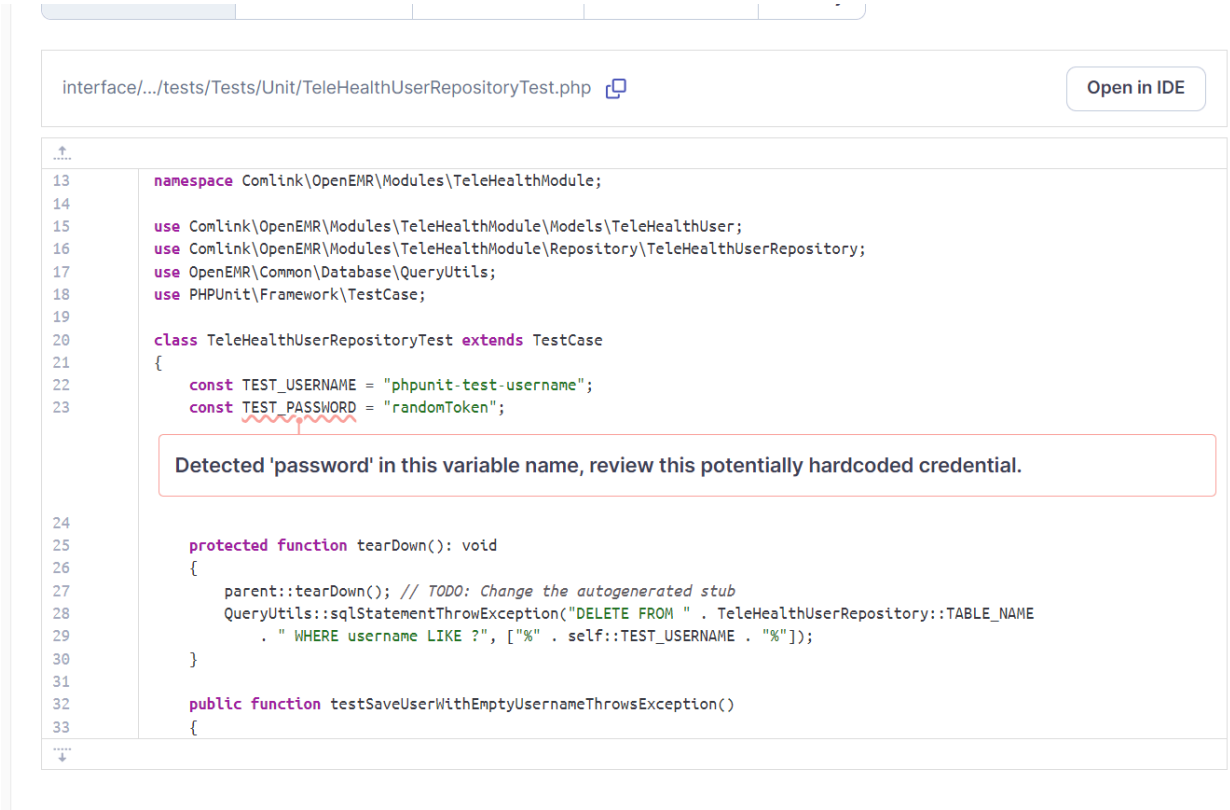
5. Insecure pseudo random generator used for sensitive certificate signing request

Cross reference: library/create_ssl_certificate.php
CWE: 338
ASVS: 6.3.1



Solution: Elevate security standards by employing a more secure random number generator for functions handling sensitive operations, such as openssl_random_pseudo_bytes(). This precautionary measure fortifies the integrity of cryptographic processes. Regularly assess and update security protocols to maintain a resilient defense against potential vulnerabilities in sensitive functions.

6. **The Detected 'password' in this variable name, review this potentially hard coded credential.**
   Cross-reference:
   interface/modules/custom_modules/oe-module-comlink-telehealth/tests/Tests/Unit/TeleHealthUserRepositoryTest.php
   CWE: 798
   ASVS: 6.4.1

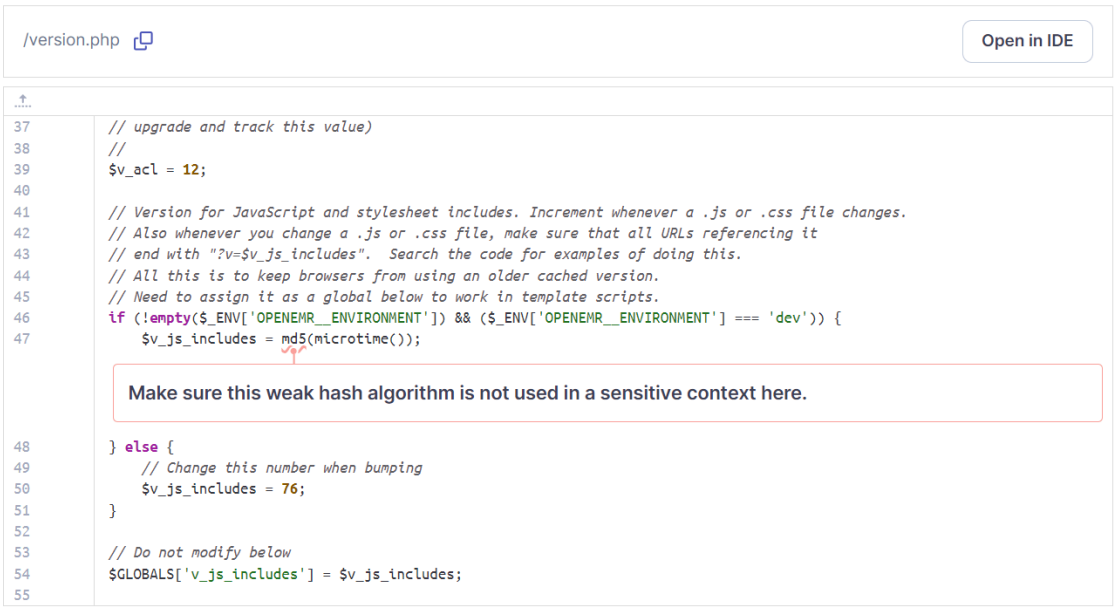


Solution : Although these credentials are hardcoded just for testing purposes, hard coded secrets in any part of the code presents a risk that should be mitigated. Therefore, it is a true positive. It can be mitigated using techniques such as using environment variables.

False Positives:

7. Make sure this weak hash algorithm is not used in a sensitive context here

Cross-reference: version.php



Explanation : This is a false positive. In this scenario, the MD5 hash is being used to generate a unique version identifier for caching purposes in a development environment, not for securing sensitive data or authentication mechanisms.

8. Detected 'password' in this variable name, review this potentially hardcoded credential.

Cross-reference: src/Common/Auth/AuthUtils.php

src/Common/Auth/AuthUtils.php Open in IDE

```
79          }
80
81          // Set up AuthHash instance (note it uses auth mode)
82          $this->authHashAuth = new AuthHash('auth');
83
84          // Ensure timing attack stuff is in place. This will be to prevent a bad actor from guessing
85          //  usernames and knowing they got a hit since the hash verification will then take time
86          //  whereas essentially no time is taken when the user does not exist. This will place
87          //  a dummy hash at $this->dummyHash, which is used by preventTimingAttack() function to
88          //  simulate a passwordVerify() run using the same hashing algorithm.
89          $dummyPassword = "dummy";
```

Detected 'password' in this variable name, review this potentially hardcoded credential.

```
90          $timing = privQuery("SELECT * FROM `globals` WHERE `gl_name` = 'hidden_auth_dummy_hash'");
91          if (empty($timing)) {
92              // Create and store a new dummy hash globals entry
93              $this->dummyHash = $this->authHashAuth->passwordHash($dummyPassword);
94              privStatement("INSERT INTO `globals` (`gl_name`, `gl_value`) VALUES ('hidden_auth_dummy_hash', ?)", [$this->
dummyHash]);
95          } elseif (empty($timing['gl_value'])) {
96              // Create and store a dummy rehash in existing globals entry
97              $this->dummyHash = $this->authHashAuth->passwordHash($dummyPassword);
98              privStatement("UPDATE `globals` SET `gl_value` = ? WHERE `gl_name` = 'hidden_auth_dummy_hash'", [$this->
dummyHash]);
99          } else {
```

Explanation : The code here uses a "dummy" password to stop attackers from guessing user names based on how long the system takes to respond. This seems like a security trick, not a mistake. The "dummy" password isn't a real secret or password; it's just a fake value used to make the system's response time consistent, whether the username exists or not.

So, This is a false positive.

9. Detected 'password' in this variable name, review this potentially hardcoded credential.

Cross-reference: src/Common/Auth/AuthUtils.php

src/Common/Auth/AuthUtils.php

Open in IDE

```
1260            error_log(
"Unable to send OpenEMR admin email notification since either patient_reminder_sender_email or
practice_return_email_path global was not set"
);
1261            return false;
1262        }
1263    }
1264
1265    // Function to prevent timing attacks
1266    //  For standard authentication, simulating a call to passwordVerify() run using the same hashing algorithm.
1267    //  For ldap authentication, simulating a call to ldap server.
1268    private function preventTimingAttack()
1269    {
1270        $dummyPassword = "heyheyhey";
```

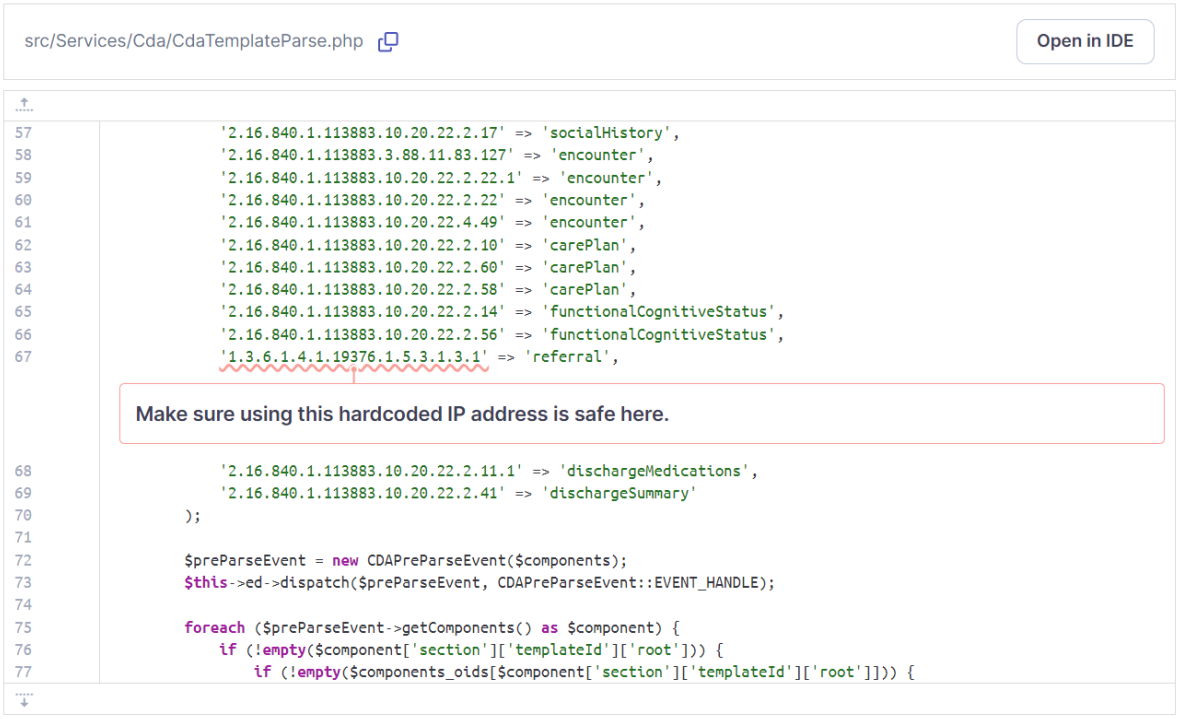
Detected 'password' in this variable name, review this potentially hardcoded credential.

```
1271        if ($GLOBALS['gbl_ldap_enabled']) {
1272            // ldap authentication simulation
1273            $this->activeDirectoryValidation("dummyCheck", $dummyPassword);
1274        } else {
1275            // standard authentication simulation
1276            AuthHash::passwordVerify($dummyPassword, $this->dummyHash);
1277        }
1278    }
1279
1280    // Function to support clearing password from memory
```

Explanation : This code is using a "dummyPassword" as part of a security measure to protect against timing attacks, similar to the previous example. The "dummyPassword" is not an actual credential but a placeholder value used to ensure the time it takes to verify a password is consistent, regardless of whether the user exists or not. This helps prevent attackers from guessing usernames based on response times.

10. Make sure using this hardcoded IP address is safe here.

Cross-reference: src/Services/Cda/CdaTemplateParse.php



Explanation : This is a false positive because SonarQube has identified it as a IP address while it is not even in the format of IP address. The strings used are identifiers for types of clinical data or components within a CDA document, not IP addresses.

Second Part Report:

Total time : 14 hrs
True Positive Vulnerabilities: 6
True Positive Vulnerabilities per hour : 6/14 = 0.429