

## 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

The screenshot shows a static application security tool interface. At the top, the file path 'modules/sms\_email\_reminder/sms\_clickatell.php' is displayed next to a copy icon. To the right is a button labeled 'Open in IDE'. Below this, a code editor displays PHP code. Line 99 is highlighted, showing the variable assignment: `var $curl_proxyuserpwd = "login:secretpass";`. A red dot is placed on the 'pwd' part of the variable name. A red-bordered box with a light purple background contains the message: 'Detected 'pwd' in this variable name, review this potentially hardcoded credential.' The code editor also shows other parts of the file, including comments for proxy settings and a callback function.

```
89  /**
90   * Proxy URL and PORT
91   * @var mixed
92   */
93   var $curl_proxy = "http://127.0.0.1:8080";
94
95   /**
96   * Proxy username and password
97   * @var mixed
98   */
99   var $curl_proxyuserpwd = "login:secretpass";
100
101  /**
102   * Callback
103   * 0 - Off
104   * 1 - Returns only intermediate statuses
105   * 2 - Returns only final statuses
106   * 3 - Returns both intermediate and final statuses
107   * @var integer
108   */
109   var $callback = 0;
```


**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

docker/library/dockers/dev-nginx/Dockerfile 

Open in IDE

```
2  # Copyright (C) 2016 Brady Mitter <brady.g.mitter@gmail.com>
3  #
4  # This program is free software; you can redistribute it and/or modify
5  # it under the terms of the GNU General Public License as published by
6  # the Free Software Foundation; either version 3 of the License, or
7  # (at your option) any later version.
8  #
9  # php-fpm Dockerfile build for openemr development docker environment
10 # This docker is hosted here: https://hub.docker.com/r/openemr/dev-php-fpm/ <tag is 7.2>
11 #
12 FROM nginx
13
14 # Copy over the nginx.conf conf
15 COPY nginx.conf /etc/nginx/nginx.conf
16
17 # Copy over the dummy self signed key/cert
18 COPY dummy-cert /etc/nginx/dummy-cert
19 COPY dummy-key /etc/nginx/dummy-key
20
21 # Needed to ensure permissions work across shared volumes with openemr, nginx, and php-fpm dockers
22 RUN usermod -u 1000 nginx
```

The nginx image runs with root as the default user. Make sure it is safe here.


**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

custom/export\_labworks.php 

Open in IDE

```
280 @unlink("$EXPORT_PATH/PM1%08.0T.DEM", $nextnumber - 5);
281 }
282
283 // End of serialized code.
284 rename("$EXPORT_PATH/locked", "$EXPORT_PATH/unlocked");
285
286 // If we have an ftp server, send it there and then rename it.
287 if ($FTP_SERVER) {
288     $ftpconn = ftp_connect($FTP_SERVER) or die("FTP connection failed");
289
290     ftp_login($ftpconn, $FTP_USER, $FTP_PASS) or die("FTP login failed");
291     if ($FTP_DIR) {
292         ftp_chdir($ftpconn, $FTP_DIR) or die("FTP chdir failed");
293     }
294     ftp_put($ftpconn, $initialname, $finalpath, FTP_BINARY) or die("FTP put failed");
295     ftp_rename($ftpconn, $initialname, $finalname) or die("FTP rename failed");
296     ftp_close($ftpconn);
297 }
298 ?>
```

Using ftp\_connect() is insecure. Use ftp\_ssl\_connect() instead

**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

Centrally managed impacted: **Security**

Introduced 8 years ago

☐ Open ☐ Not assigned ☒ Vulnerability ☐ Critical

Where is the issue?	Why is this an issue?	How can I fix it?	Activity	More Info
---------------------	-----------------------	-------------------	----------	-----------

```
99  sam.li...      $data = array('RxInput' => $xml);
100
101               curl_setopt($curlHandler, CURLOPT_URL, $this->getGlobals()->getPath());
102               curl_setopt($curlHandler, CURLOPT_POST, 1);
103               curl_setopt($curlHandler, CURLOPT_POSTFIELDS, 'RxInput=' . $xml);
104               curl_setopt($curlHandler, CURLOPT_SSL_VERIFYPEER, 0);
```

Enable server certificate validation on this SSL/TLS connection.

Embedded database should be used for evaluation purposes only

Embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by SonarSource SA


**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

library/create\_ssl\_certificate.php 

Open in IDE

```
86
87  /**
88   * Create a certificate, signed by the given Certificate Authority.
89   * @param $csr      - The certificate signing request
90   * @param $cacert    - The Certificate Authority to sign with, or NULL if not used.
91   * @param $cakey     - The Certificate Authority private key data to sign with.
92   * @return data     - A signed certificate, or false on error.
93   */
94  function create_crt($csr, $cacert, $cakey)
95  {
96      $cert = openssl_csr_sign($csr, $cacert, $cakey, 3650, ['digest_alg' => 'sha256'],
rand(1000, 9999));
97
98      return $cert;
99  }
100
101  /**
102   * Create a new client certificate for a username or client hostname.
```

Make sure that using this pseudorandom number generator is safe here.

**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

interface/.../tests/Tests/Unit/TeleHealthUserRepositoryTest.php 

```
13 namespace Comlink\OpenEMR\Modules\TeleHealthModule;
14
15 use Comlink\OpenEMR\Modules\TeleHealthModule\Models\TeleHealthUser;
16 use Comlink\OpenEMR\Modules\TeleHealthModule\Repository\TeleHealthUserRepository;
17 use OpenEMR\Common\Database\QueryUtils;
18 use PHPUnit\Framework\TestCase;
19
20 class TeleHealthUserRepositoryTest extends TestCase
21 {
22     const TEST_USERNAME = "phpunit-test-username";
23     const TEST_PASSWORD = "randomToken";
24
25     protected function tearDown(): void
26     {
27         parent::tearDown(); // TODO: Change the autogenerated stub
28         QueryUtils::sqlStatementThrowException("DELETE FROM " . TeleHealthUserRepository::TABLE_NAME
29             . " WHERE username LIKE ?", ["%" . self::TEST_USERNAME . "%"]);
30     }
31
32     public function testSaveUserWithEmptyUsernameThrowsException()
33     {
34     }
```


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

**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

/version.php 

Open in IDE

```
37 // upgrade and track this value)
38 //
39 $v_acl = 12;
40
41 // Version for JavaScript and stylesheet includes. Increment whenever a .js or .css file changes.
42 // Also whenever you change a .js or .css file, make sure that all URLs referencing it
43 // end with "?v=$v_js_includes". Search the code for examples of doing this.
44 // All this is to keep browsers from using an older cached version.
45 // Need to assign it as a global below to work in template scripts.
46 if (!empty($_ENV['OPENEMR_ENVIRONMENT']) && ($_ENV['OPENEMR_ENVIRONMENT'] === 'dev')) {
47     $v_js_includes = md5(microtime());
48 } else {
49     // Change this number when bumping
50     $v_js_includes = 76;
51 }
52
53 // Do not modify below
54 $GLOBALS['v_js_includes'] = $v_js_includes;
55
```


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

**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";
90
91     $timing = privQuery("SELECT * FROM `globals` WHERE `gl_name` = 'hidden_auth_dummy_hash'");
92     if (empty($timing)) {
93         // Create and store a new dummy hash globals entry
94         $this->dummyHash = $this->authHashAuth->passwordHash($dummyPassword);
95         privStatement("INSERT INTO `globals` (`gl_name`, `gl_value`) VALUES ('hidden_auth_dummy_hash', ?)", [$this->
96         dummyHash]);
97     } elseif (empty($timing['gl_value'])) {
98         // Create and store a dummy rehash in existing globals entry
99         $this->dummyHash = $this->authHashAuth->passwordHash($dummyPassword);
100        privStatement("UPDATE `globals` SET `gl_value` = ? WHERE `gl_name` = 'hidden_auth_dummy_hash'", [$this->
101        dummyHash]);
102    } else {
103    }
```

**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     $dummyPassword = "heyheyhey";  
1271  
1272     if ($GLOBALS['gbl_ldap_enabled']) {  
1273         // ldap authentication simulation  
1274         $this->activeDirectoryValidation("dummyCheck", $dummyPassword);  
1275     } else {  
1276         // standard authentication simulation  
1277         AuthHash::passwordVerify($dummyPassword, $this->dummyHash);  
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

src/Services/Cda/CdaTemplateParse.php 

Open in IDE

```
57         '2.16.840.1.113883.10.20.22.2.17' => 'socialHistory',
58         '2.16.840.1.113883.3.88.11.83.127' => 'encounter',
59         '2.16.840.1.113883.10.20.22.2.22.1' => 'encounter',
60         '2.16.840.1.113883.10.20.22.2.22' => 'encounter',
61         '2.16.840.1.113883.10.20.22.4.49' => 'encounter',
62         '2.16.840.1.113883.10.20.22.2.10' => 'carePlan',
63         '2.16.840.1.113883.10.20.22.2.60' => 'carePlan',
64         '2.16.840.1.113883.10.20.22.2.58' => 'carePlan',
65         '2.16.840.1.113883.10.20.22.2.14' => 'functionalCognitiveStatus',
66         '2.16.840.1.113883.10.20.22.2.56' => 'functionalCognitiveStatus',
67         '1.3.6.1.4.1.19376.1.5.3.1.3.1' => 'referral',

68         '2.16.840.1.113883.10.20.22.2.11.1' => 'dischargeMedications',
69         '2.16.840.1.113883.10.20.22.2.41' => 'dischargeSummary'
70     );
71
72     $preParseEvent = new CDAPreParseEvent($components);
73     $this->ed->dispatch($preParseEvent, CDAPreParseEvent::EVENT_HANDLE);
74
75     foreach ($preParseEvent->getComponents() as $component) {
76         if (!empty($component['section']['templateId']['root'])) {
77             if (!empty($components_oids[$component['section']['templateId']['root']])) {
```

Make sure using this hardcoded IP address is safe here.

**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 :  $6/14 = 0.429$