Report on CVE-2020-11476

By Trent Chismar, Ryan Brockway, Angela Castronuovo, Joshua Cominelli

Short Description

Concrete5 is a free, open source content management system that allows users with little technical experience to easily build and maintain their own website. With concrete5, users are able to fully design and customize their website without needing to write any code. One aspect of the website that the user does not have full control over is what files can be uploaded to the site that is built using concrete5. Concrete5 blocks files with certain file extensions from being uploaded to the site. However, concrete5 did not block every possible file extension that an attacker could use to perform an attack using concrete5. Concrete5 before version 8.5.3 blocked files with the extension .php and .phtml, but the application did not block the file extension .phar, which is a PHP archive file. This is a vulnerability, as web servers interpret phar files as php files. As a result, an attacker is able to bypass the .php file extension restriction by simply uploading a phar file. An attacker is then able to upload and remotely execute PHP over the network. This vulnerability was discovered on March 31, 2020, and was eventually fixed by concrete5 on June 4, 2020.[1]

Who can attack?

Anyone with their own website that was built using concrete5 is able to perform this attack. This attack requires the attacker to have high privileges. This is because the only way for an attacker to execute this attack is if they have admin privileges for the given concrete5 website. This is because the attacker has to specify in the “Allowed File Types” section of the website what files extensions are allowed to be uploaded to the site. The attacker would add the .phar file extension to this list, so they are then able to upload a phar file in order to carry out the attack. This attack can only be executed by an attacker who has access to the concrete5 admin tools for that specific website. Therefore, anyone who has created their own website using concrete5 and has basic PHP knowledge is able to perform this attack. Concrete5 is a free to use content management system, which gives anyone the ability to create their own website to perform this attack.[2]

[1] https://herolab.usd.de/security-advisories/usd-2020-0041/
[2] https://herolab.usd.de/security-advisories/usd-2020-0041/

Why does the attack matter?

The significance of this attack comes from the fact that concrete5 is used for creating websites. So the risk of this attack extends beyond the application itself, rather any website using concrete5 is exploitable. An e-commerce website could risk its users' financial information. A health website could risk medical records. There are over 700,000 websites online with concrete5, including the US Army, the German multinational chemical company BASF, and the Renewable Energy Corporation.[3]

Damages Caused:

This attack is a Remote Code Execution vulnerability meaning that the concrete5 website allowed arbitrary code to be triggered and executed over a network.[4] After a file of PHP code is uploaded to the concrete5 website, the attacker is given a lot of control over the user's server and computer. The attacker can install spyware onto the user's computer or even gain root access to the user's computer system which would give the attacker continuous remote access. This is particularly concerning because the attacker can steal personal or financial data and sometimes, if the attack is discrete enough, without even being detected. Since any user who clicks on the compromised website will be infected, the attack has the potential to reach large numbers of users and its impact can be widespread.

How does it work?

To perform a remote code execution attack on a concrete5 server, the adversary would create a file with an extension such as php, phar, php8, shtml, etc. The adversary must explicitly whitelist the file type being used, which requires administrative permissions on the website. This file could contain the following PHP code, which runs a command provided by the adversary:

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

This file is then uploaded to the server via concrete5. From here, viewing the "properties" of the file provides a URL to it like this:

```
http://localhost/application/files/4015/8558/7320/shell.phar
```

[3] https://www.concrete5.org/
[4] https://www.malwarebytes.com/backdoor/

By altering the URL, the adversary can now pass the “cmd” parameter to the PHP code, and upon opening the URL, the file will automatically execute and run the command. For example, the adversary could execute the UNIX command “whoami” by opening the the following URL:

```
http://localhost/application/files/4015/8558/7320/shell.phar?cmd=whoami
```

At this point, any terminal command is possible. The adversary has full control over the server, and can view or alter files, modify privileges, or upload more files with malicious code. Executing a command is as simple as creating a new URL based on the uploaded phar file and opening it.[5]

Fixes

Concrete5 fixed the vulnerability in an update, version 8.5.3, released on June 4th. They added a hard coded blacklist of file extensions to keep php files from being uploaded.[6] Interestingly, the people who found the vulnerability think that this is an insufficient fix. Since there are many different file extensions that can be interpreted as php code, the security researchers think instead of adding all those file types to a blacklist, it would be more efficient to make a whitelist of the relatively few valid extensions.[7]

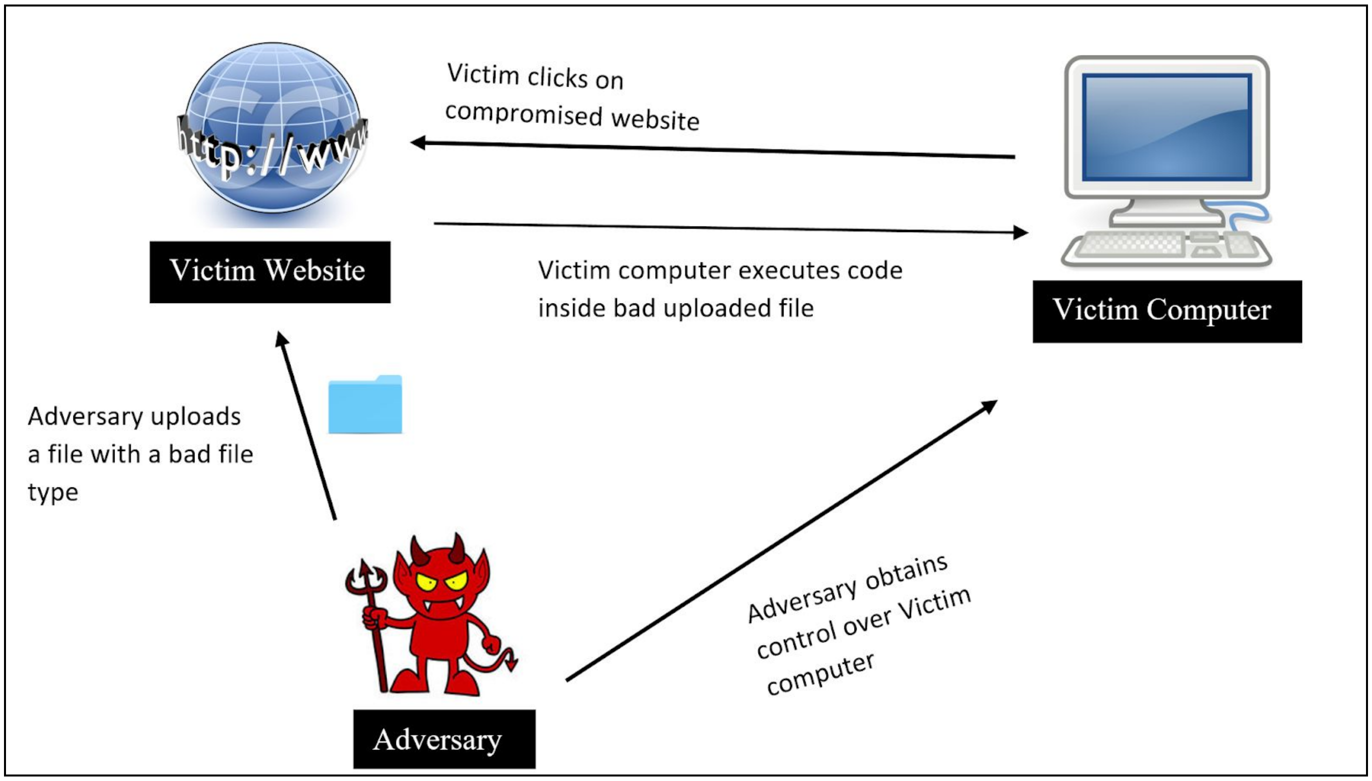
For users who did not want to update to the newest version, Concrete5 said to change the setting to keep uploaded files of any extension type from running. This fix is not viable in the long run, since even files like html or jpeg would be prevented from running. This is why concrete5 strongly recommends everyone to make sure they are running the last version.

[5] https://herolab.usd.de/security-advisories/usd-2020-0041/

[6]https://www.concrete5.org/about/blog/security/press-articles-about-rce-file-upload-vulnerability-cve-2020-11476-friendly-reminder-update-version-854-security-and-bug-fixes

[7] https://herolab.usd.de/security-advisories/usd-2020-0041/

Threat Model:


Victim clicks on compromised website
Victim Website
Victim computer executes code inside bad uploaded file
Victim Computer
Adversary uploads a file with a bad file type
Adversary obtains control over Victim computer
Adversary

The victim website, a website builder, contains a file upload form without an extensive file type blacklist.[8] This means it allows file types that can be used in a malicious manner such as .php, .phar, and .htaccess extensions. Because of this, the adversary can exploit this vulnerability by using this victim website to create his/her own website and can upload a .php file containing malicious PHP code to the file upload form so that the code will be run by the web server.

Once an unsuspecting user clicks on the adversary's compromised website, the PHP code inside the uploaded file will be executed on the user's computer. The adversary can then gain full control of the user's computer and server by injecting reverse-shell[9] so that he/she can control the cmd remotely. The adversary can also install malware to the victim's computer and can potentially access sensitive data.

CVE vector justification

Vector: CVSS:3.1/ **AV:N /AC:L /PR:H /UI:N /S:U /C:H /I:H /A:H**[10]

[8] https://www.acunetix.com/websitesecurity/upload-forms-threat/
[9] https://hackerone.com/reports/768322
[10] https://nvd.nist.gov/vuln/detail/CVE-2020-11476

Attack Vector (AV: Network)

We agree with this assessment of risk. Since all the attacker has to do is add a file extension to the “Allow File types” field in the website’s settings, this can be done from any machine hooked up to the internet.

Attack Complexity (AC: Low)

Changing the allowed file types is as simple as adding the extension to a text field. The only aspect of this attack that requires any background knowledge is the PHP script, but even that is nothing requiring extremely specialized knowledge, meaning Low is an accurate Attack Complexity rating.

Privileges required (PR : High)

This is an accurate assessment of required privilege. The entire attack relies on the attacker being able to change the list of allowed file types in the settings page. This requires admin access which is not available to a typical visitor of the site.

User interaction (UI : None)

The level of user interaction is accurate. Assuming the attacker has high enough access to change the file types in settings, this whole attack can be executed without involving any user input.

Scope (S : Unchanged)

We disagree with the assessment that the scope is unchanged, and instead we think it fits the criteria for a changed rating. Here the vulnerable component is a Concrete5 website. However, since the attacker has full control over the server, he can gain information beyond the site itself, like user data, for example credit card info, passwords and usernames. Furthermore, since this vulnerability allows the attacker to insert any PHP of his choosing onto the server, there is the possibility of him creating malware to infect the computers of the site’s users.

Confidentiality impact (C : High)

High is an accurate assessment of the Confidentiality Impact. Once the attacker changes the allowed file types and uploads his PHP code, he has full control over the site. This means he has full access to any data stored on the web server.

Integrity impact (I : High)

Since the PHP code gives the attacker full control over the site, the integrity impact is very high.

Availability impact (A : High)

The rating of High is accurate because since the attacker has full control over the web server, he can control legitimate users’ access to the site.