Chloe Chamberland

August 4, 2020

Critical Vulnerability Exposes over 700,000 Sites Using Divi, Extra, and Divi Builder

On July 23, 2020, our Threat Intelligence team discovered a vulnerability present in two themes by Elegant Themes, Divi and Extra, as well as Divi Builder, a WordPress plugin. Combined, these products are installed on an estimated 700,000 sites. This flaw gave authenticated attackers, with contributor-level or above capabilities, the ability to upload arbitrary files, including PHP files, and achieve remote code execution on a vulnerable site's server.

We initially reached out to Elegant Themes on July 23, 2020 and, after establishing an appropriate communication channel, we provided the full disclosure details on July 28, 2020. The developers responded on June 29, 2020 to let us know a patch would be coming in the next version. Patches were released yesterday, on August 3, 2020, in version 4.5.3 for all products.

This is considered a critical security issue that could lead to remote code execution on a vulnerable site's server. If you haven't already updated, and you are running Divi versions 3.0 and above, Extra versions 2.0 and above, or Divi Builder versions 2.0 and above, we highly recommend updating to the patched version, 4.5.3 , immediately. Alternatively, you can use their Security Patcher Plugin until you can update safely.

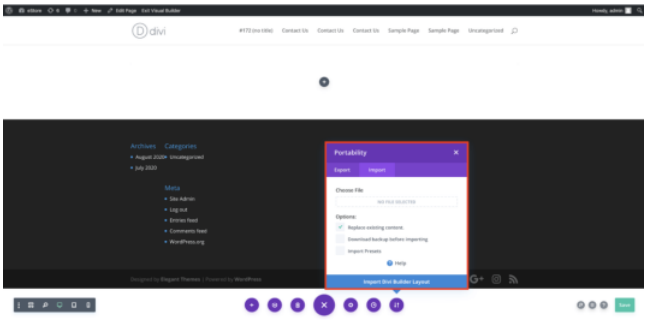
Both Wordfence Premium and free users are protected against any attacks attempting to exploit this vulnerability due to the Wordfence firewall's built-in malicious file upload protection.

Description: Authenticated Arbitrary File Upload
Affected Products: Divi Theme, Extra Theme, and Divi Builder plugin
Theme Slugs: divi, extra
Plugin slug: divi-builder
Affected Versions: (Divi): 3.0 – 4.5.2
Affected Versions: (Extra): 2.0 – 4.5.2
Affected Versions: (Divi Builder): 2.0 – 4.5.2
CVE ID: CVE-2020-35945
CVSS Score: 9.9 (CRITICAL)
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H
Fully Patched Version (same for all products): 4.5.3

Elegant Themes is the creator behind one of the most popular premium themes, Divi. One of the features of the Divi theme is that it comes with the Divi Page Builder that makes the site design and editing process easy and customizable. In addition to the Divi theme, Elegant Themes offers an alternative theme, Extra, that includes the Divi Builder. The standalone Divi Builder plugin is also available and can be used with any theme.

As part of the Divi Builder functionality, users that have the ability to create posts can import and export Divi page templates using the portability feature.

Unfortunately, we discovered that although this feature used a client-side file type verification check, it was missing a server-side verification check. This flaw made it possible for authenticated attackers to easily bypass the JavaScript client-side check and upload malicious PHP files to a targeted website. An attacker could easily use a malicious file uploaded via this method to completely take over a site.



Divi Builder portability feature used to import layouts

What went wrong?

Taking a closer look at the code, we can see that the portability `import` function was triggered with the use of the `et_core_portability_import` AJAX action and corresponding `et_core_portability_ajax_import` function, which does have nonce and capability check.

```
2324 | add_action( 'wp_ajax_et_core_portability_import', 'et_core_portability_ajax_import' );
```

The core of the problematic code could be found within the `import` function of the builder's `portability.php` file. Since the plugin had a client-side JavaScript-based file extension check for .json files, the developers might have missed adding a server-side file-type check here prior to using the file's contents during the import, or assumed the client-side check would be sufficient protection.

```
public function import( $file_context = 'upload' ) {
    global $shortname;


    $timestamp              = $this->get_timestamp();
    $filesystem             = $this->set_filesystem();
    $temp_file_id           = sanitize_file_name( $timestamp );
    $temp_file              = $this->has_temp_file( $temp_file_id, 'et_core_import' );
    $include_global_presets = isset( $_POST['include_global_presets'] ) ? wp_validate_boolean( $_POST['include_global_
    $global_presets         = '';

    if ( $temp_file ) {
        $import = json_decode( $filesystem->get_contents( $temp_file ), true );
    } else {
        if ( ! isset( $_FILES['file'] ) ) {
            return false;
        }

        if ( ! in_array( $file_context, array( 'upload', 'sideload' ) ) ) {
            $file_context = 'upload';
        }
```

Analyzing the code further, we see that the file is temporarily uploaded using `wp_handle_upload`, with `test_type` set to false, overriding the `wp_check_filetype_and_ext` function that checks a file's type and determines if it is a safe file to upload based on a list of allowed mime types.

This meant that the `wp_handle_upload` function did not test the file type during the upload, essentially disabling the extensive file-type checking protection built-in to the function.

```
$handle_file = "wp_handle_{$file_context}";
$upload      = $handle_file( $_FILES['file'], array(
    'test_size' => false,
    'test_type' => false,
    'test_form' => false,
) );
```

…

```
do_action( 'et_core_portability_import_file', $upload['file'] );
```

From there, the file's content was checked to see if it could be used for the import. If the file's content did not appear to be usable JSON data for an import, then the process was killed and the message `'importContextFail'` was returned.

```
$temp_file = $this->temp_file( $temp_file_id, 'et_core_import', $upload['file'] );
$import = json_decode( $filesystem->get_contents( $temp_file ), true );
$import = $this->validate( $import );
$import['data'] = $this->apply_query( $import['data'], 'set' );

if ( ! isset( $import['context'] ) || ( isset( $import['context'] ) && $import['context'] !== $this->instance->contex
    return array( 'message' => 'importContextFail' );
}
```

Toward the end of the function, there was a hook to the function `'delete_temp_files'` that was intended to delete any JSON files used for the import once completed. However, since the import died for files without usable JSON content before getting to this function, the files remained in the uploads directory until a legitimate JSON file was imported.

```
$this->delete_temp_files( 'et_core_import' );
```

This flaw made it possible for authenticated users with the `edit_posts` capability, like contributors, editors, and authors, to upload arbitrary files. An attacker could easily upload malicious PHP files and access them from the uploads directory. This could ultimately result in remote code execution and complete compromise of a vulnerable site's hosting account.

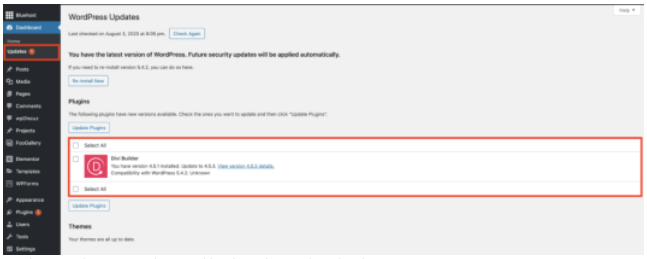
The `wp_ajax_et_theme_builder_api_import_theme_builder` AJAX action and corresponding function used to import a theme builder template was also susceptible to arbitrary file uploads due to the same issues, however, exploiting this would have required administrative privileges thus significantly reducing it's severity.

Fortunately, Elegant Themes was very quick to respond and release a patch that not only prevented all files except .json files from being uploaded, but also ensured that files would be sufficiently deleted at any stage of the process once no longer used.

How to Update your Elegant Themes Product

As long as you have supplied your Elegant Themes Username and API key on your WordPress site, then you can take care of your updates directly in the updates area on your site. To do so, log into your site, and navigate to the "Updates" area. Select the Elegant Themes product you would like to update and just click "Update Plugin" or "Update Theme" depending on which product you are updating.

Also, please note that Elegant Themes has made this patch available to users, even if your account is expired.



WordPress updates area with Divi Builder plugin that needs updated.

If you are unable to update fully, you can install Elegant Themes Security Patcher Plugin that will temporarily patch the vulnerability until you are able to do a complete update.

Another way to stay protected

As mentioned, in our post last week, Wordfence has a feature to disable code execution in the uploads directory. Even if you're not using one of Elegant Themes' vulnerable products, we highly recommend enabling this setting as it will provide additional protection against vulnerabilities like this one that may erroneously allow PHP files to be uploaded into the uploads directory.

With this option enabled, attackers will not be able to execute PHP files uploaded into the uploads directory, providing an extra layer of security and assisting in thwarting attacks like this one. In the event that a zero-day vulnerability is discovered and actively exploited prior to the creation of a custom firewall rule, having this feature enabled can help keep your site protected.

The 'Disable Code Execution for Uploads directory' option location.

Proof of Concept Walkthrough

Due to the critical severity of this vulnerability and high user install base, we are refraining from posting a proof of concept walkthrough video for this vulnerability at this time. If you are interested to learn how this vulnerability might be exploited, please join us for Wordfence Office Hours next week on Tuesday, August 11th at 12:00 EST. This allows us to give you time to update and still provide you with the in-depth details on how this could have been exploited on unprotected sites.

Disclosure Timeline

Jul 23, 2020 – Initial discovery of vulnerability. We verify the Wordfence firewall provides protection against exploit attempts and we make our initial contact attempt with the Elegant Themes team.
July 27, 2020 – The developer confirms inbox for handling disclosure.
July 28, 2020 – We send full disclosure details.
July 28, 2020 – They respond letting us know they have begun working on a patch and anticipate releasing it on the upcoming Monday.
July 31, 2020 – They send us the details of the patch so we can verify the fix is sufficient.
August 3, 2020 – A patch is released in version 4.5.3 for all products.

Conclusion

In today's post, we detailed a flaw in Elegant Themes' products Divi, Extra, and Divi Builder that provided authenticated users with the ability to upload arbitrary files, including PHP files, and execute any code in those files on the server. This flaw has been fully patched in version 4.5.3 for all products. We recommend that users immediately update to the latest version available, which is version 4.5.3 at the time of this publication.

Sites using Wordfence Premium as well as those still using the free version of Wordfence are protected from attacks against this vulnerability. If you know a friend or colleague who is using one of these themes or the plugin on their site, we highly recommend forwarding this advisory to them to help keep their sites protected as this is a critical security update.

Special thanks to Mitch, from Elegant Themes, for working with us to quickly get a patch out to protect Elegant Themes users.

Comments

11 Comments

Christopher Davis *
August 4, 2020
6:46 am

Can you please describe how: "Sites using Wordfence Premium as well as those still using the free version of Wordfence are protected from attacks against this vulnerability."

I believe the Disable Code Execution for Uploads directory is off by default. Is Wordfence free version protecting sites in another way from this vulnerability?

Thanks
Chris

Chloe Chamberland *
August 4, 2020
7:00 am

Hi Chris!

The Wordfence Firewall comes with some standard baseline firewall rules to block common exploit attempts that target vulnerabilities like XSS, SQLi, LFI, etc... Both free and premium users have these rules. One of these standard rules is a malicious file upload rule that will block any malicious files, like PHP files, from being uploaded. That firewall rule is what is keeping both free and premium users safe from any exploit attempts targeting this vulnerability as it will block any attempt to upload a malicious file.

Hope that helps! Let me know if you have further questions.

I Love Wordfence *
August 4, 2020
7:35 am

Thanks team!
I added the extra tip for the upload directory as well, thanks for keeping us safe :)

Kathir *
August 4, 2020
7:45 am

Thanks for mentioning vulnerability. now only i got to know about it, thank god nothing happened.. now i updated...

Lynne *
August 4, 2020
8:08 am

Hi! All this talk is foreign to me, but I appreciate the help. But in order to be safe from this attack, am I supposed to UNCHECK the 'Disable Code Execution for Uploads directory' option , or keep it as is? I'm confused about that. Thank you!

Chloe Chamberland *
August 4, 2020
10:36 am

Hi Lynne!

You should leave the option checked in order to disable code execution for uploads directory. I want to note that your site will be safe from any exploit attempts targeting this vulnerability, even without 'Disable Code Execution for Uploads directory' feature

Lynne *
August 4, 2020
12:01 pm

I will keep it checked. Thank you!!

Steve *
August 4, 2020
8:30 am

Nice work, I struggle to comprehend how this type of vulnerability is not double checked at theme creation, even on multi purpose themes with all its functionality, especially on such a popular theme.

I guess as its based on 'pre-approved' User access. Time to review contributor logins ! 😁

Catarina *
August 4, 2020
11:24 am

Is this vulnerability isolated to these role types only? Assuming if you don't have any or the ability for people to join as a contributor or above that you aren't exposed to the vulnerability?

Chloe Chamberland *
August 4, 2020
11:47 am

Hi Catarina!

That is correct - if you do not have any untrusted users with the edit_posts capability (i.e. contributors, authors, and editors) then you should be relatively unexposed as this vulnerability requires that capability to exploit. I should note that this can *technically* be exploited by administrative users, though that would requiring gaining access to the administrative account, so as always just make sure you are using a strong password for your administrative account(s).

And you will be completely unexposed to this vulnerability if you update the plugin/themes to the latest version (4.5.3) that contains the patch. :)

Hope that helps!

Catarina *
August 4, 2020
11:50 am

Thank you, that's very clear and thank you for getting back so quickly.