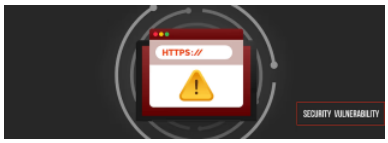
CVE-2020-27304 – RCE via Directory Traversal in CivetWeb HTTP server

By Denys Vozniuk and Shachar Menashe | October 19, 2021
6 min read



Background

JFrog has recently disclosed a directory traversal issue in CivetWeb, a very popular embeddable web server/library that can either be used as a standalone web server or included as a library to add web server functionality to an existing application. The issue has been assigned to CVE-2020-27304.

This directory traversal issue is highly exploitable and can lead to remote code execution, especially if the web server is running as root – due to the attacker's ability to add or overwrite files that are subsequently executed.

Who is actually impacted?

This issue affects CivetWeb versions 1.8 to 1.14 (inclusive), and was recently fixed with the 1.15 release.

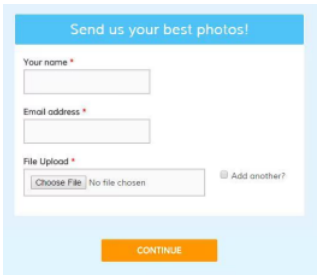
This issue only impacts CivetWeb-based web applications that use the built-in file upload form handler.
In technical terms, a CivetWeb-based web application is vulnerable if:

1. The application handles HTTP form data by calling CivetWeb's `handle_form_request` and supplies the (mandatory) user-defined `field_found` callback function
2. The `field_found` callback function returns `MG_FORM_FIELD_STORAGE_STORE` to indicate a file upload operation
3. The `field_found` callback function supplies the (mandatory) path output argument, where the path relies on the filename input argument (which comes directly from the HTTP form data)

Note that this scenario is the standard way of using CivetWeb's file upload functionality, and is supplied as a full working example in the "embedded_c" example in the CivetWeb sources.

CivetWeb's built-in file upload functionality

Web servers that allow HTTP clients to upload files to the server, often choose to implement this functionality using form-based file upload (RFC 1867), which usually looks like this on the client (web browser) side:



The CivetWeb server contains built-in support for this kind of file upload, via the mg_handle_form_request API.

A developer that wants file-upload support in his/her web service can simply invoke this API with a callback function that returns the `MG_FORM_FIELD_STORAGE_STORE` constant:

```
struct mg_form_data_handler fdh = {field_found_callback, field_get_callback,
field_stored_callback, 0};
...
mg_handle_form_request(conn, &fdh);
```

Example of a callback function snippet:

```
int
field_found_callback (const char *key,
              const char *filename,
              char *path,
              size_t pathlen,
              void *user_data)
{
    ...
    snprintf(path, pathlen, "/tmp/%s", filename);
    return MG_FORM_FIELD_STORAGE_STORE;
    ...
}
```

The path traversal issue

The path traversal issue's root cause is actually a missing validation for Linux-based builds of CivetWeb.

The relevant source code in CivetWeb's `mg_handle_form_request` that takes care of the HTTP request is as follows:

```
        }
        file_size = 0;
        if (fstore.access.fp != NULL) {
            size_t n = (size_t)fwrite(val, 1, (size_t)vallen,
fstore.access.fp);
    ...
}
```

The `mg_fopen` function, which is responsible for creating the uploaded file, tries to prevent path traversal attacks by calling the `mg_path_suspicious` function, but this function only checks the path separator for Windows builds:

```
/* Reject files with special characters */
static int
mg_path_suspicious(const struct mg_connection *conn, const char *path)
{
    const uint8_t *c = (const uint8_t *)path;
    ...
    while (*c) {
        if ((*c == '>') || (*c == '<') || (*c == '|')) {
            /* stdin/stdout redirection character */
                return 0;
            }
#if defined(_WIN32)
            if (*c == '\\') {
                /* Windows backslash */
                return 0;
            }
#else
            if (*c == '/') {
                /* Linux ampersand */
                return 0;
            }
        ...
```

Therefore – when CivetWeb is compiled for Linux or OSX, there is no path traversal sanitization at all for uploaded filenames.

Seeing that the "embedded_c" web service example supplied in the source repository is susceptible to this issue, this vulnerability is likely to be exploitable in CivetWeb instances that support file uploads.

Fixing the issue

The CivetWeb maintainers fixed this issue in two separate ways –

1. The form-handling code will now canonicalize the filename (before it is given to the `field_found` user callback) by removing dot segments, as defined in RFC 3986
2. The "embedded_c" example has been updated to show how path separator characters (/ or \ depending on the platform) should be filtered out, as defined in RFC 7578

We applaud CivetWeb's maintainers for fixing the issue in the most professional manner – by closely following the RFCs for HTTP forms and URIs. This is usually the best practice, and as can be seen here, it renders the implementation much more resistant to path traversal attacks. We recommend other OSS implementers to adhere to any existing relevant RFCs or alternatively – use an external library that conforms to these RFCs.

Automated detection of affected artifacts

Automated vulnerability scanning can be used to identify artifacts that contain a vulnerable version of CivetWeb. Further contextual analysis can determine the CVE's applicability in each scanned artifact — that is, if CVE-2020-27304 is actually exploitable.

In high-level terms, the contextual analyzer would need to perform the following steps to determine susceptibility to CVE-2020-27304:

1. Detect all calls to the exported API `mg_handle_form_request`
2. Analyze all callback functions that were specified as the 2nd argument to `mg_handle_form_request` (the `field_found` callback functions)
3. Check if any of the callback functions write data to the path output argument, where the data is tainted (w.r.t data flow analysis) by the filename input argument
4. Verify that the user did not implement his own path-traversal-filtering mechanism on the `filename` input argument

Conclusion and Acknowledgements

To conclude, we highly recommend anyone that is using a web library or designing his/her own web library to take path traversal attacks into consideration, and sanitize any file names coming from potential user input (or better still – implement according to the relevant RFCs).

We would like to thank CivetWeb's maintainers, for validating and fixing the issue in a short amount of time and in a very thorough manner.

In addition to exposing new security vulnerabilities and threats, JFrog provides developers and security teams easy access to the latest relevant information for their software with automated security scanning by JFrog Xray. Keep following us for product updates including automated contextual applicability analysis for Critical Vulnerabilities Exposure (CVE), allowing developers to save time and effort by fixing only the issues that have a real-life security impact.

Tags: security-research, vulnerability disclosure

Brand Guidelines

Follow Us