CVE-2021-41773

George-Andrei IOSIF[1]
[1]*Politehnica University of Bucharest, Computer Science Department, Romania*
Email: [1]george_andrei.iosif@stud.acs.upb.ro

I. Introduction

The Apache HTTP Server (often known as `httpd`) is a free and open source web server for Linux and Windows. It was first released in 1995 and has since become the most widely used software of its kind. According to Netcraft, it was running in 24 percent of the servers of the world's busiest websites in January 2021.

Support for programming languages (Perl and PHP), TLS/SSL, IPv6 compatibility, and a reverse proxy mode that allows caching are just a few of the features.

A security advisory published in October 2021 said that the Apache HTTP Server was vulnerable to path traversal and file disclosure owing to a path normalization flaw in certain circumstances. Later, it was discovered that if the Common Gateway Interface (abbreviated CGI) module is enabled, this vulnerability (identified as CVE-2021-41773) can lead to remote code execution.

II. Vulnerability Analysis

The path normalization method used by Apache HTTP Server removes any suspicious parts of the URI that the client requests. In 2.4.49, it was unable to recognize a series of characters related to directory traversal, notably `../`, due to alterations introduced before October 2021.

A second requirement that made a web server vulnerable was the setting of the value of the `Directory` XML field to `Require all granted`. This permits the server to provide files to users outside the served files' folder (for example, HTML pages).

```
<Directory/>
    AllowOverride none
    Require all granted
</Directory>
```

In this way, `/cgi-bin/.%2e/.%2e/.%2e/.%2e/<path_to_file>` represents the path requested via `GET` by an exploit. `<path_to_file>` denotes the absolute path of the file that will be included in the response (via the path traversal vulnerability).

When the module for handling CGI scripts is enabled (by activating `mod_cgi` in Apache's configuration file), this vulnerability leads to remote code execution. By including a shell binary (for example, `/bin/bash`) with the technique described above, the CGI module will execute it. Data from requests (for example, Bash commands in a `POST` request) will be passed to `stdin`. The characters returned to `stdout` by the shell (namely the output from running the provided command) will then be redirected to the web client. In this manner, the latter will achieve a code execution.

This vulnerability allows remote code execution when the module for handling CGI scripts is activated (by activating `mod cgi` in Apache's configuration file). The CGI module will execute a shell binary (for example, `/bin/bash`) if it is included with the technique mentioned above. `stdin` will receive data from requests (for example, Bash commands in a `POST` request). The shell's output (the output from running the specified command) will be routed to the web client. In this way, the latter will be able to execute code.

```
<IfModule !mpm_prefork_module>
    LoadModule cgid_module modules/mod_cgid.so
</IfModule>
```

III. Impact

The attacker can get at least an arbitrary file read by exploiting this vulnerability. Because the web server is normally launched as a non-privileged user (for example, `daemon`), it is limited to a few files, but it may leak information (configuration files, log files, etc.) that will be useful later in the attack.

On the other side, the same user constraint limits code execution on the remote system, but it might be combined with another vulnerability in the operating system to achieve `Administrator` or `root` capabilities.

IV. Mitigations

The server's owners might easily patch this issue by updating to a newer version of Apache HTTP Server, such as 2.4.51 or higher.

Some obstacles, such as sophisticated operating procedures that prevent timely changes of production servers, can make the above difficult. In these circumstances, the security network devices that act as middleware between the vulnerable system and its clients may block requests from known attackers (such as those offered by Juniper Networks [2]) or match specific content signatures. Positive Research, for example, provided shortly after the vulnerability was disclosed two Suricata (an open source intrusion detection and prevention system) rules that match exploit attempts [9].

```
alert http any any -> any any (
    [...]
    flow: established, to_server;
    content: "%2e/";
    nocase;
    http_raw_uri;
    pcre: "/\/(\.|%2e)%2e\//Ii";
    threshold: type limit, track by_src, count 1, seconds 60;
    [...]
)
```

V. Exploitation in the Wild

CVE-2021-41773 was a 0-day vulnerability that was exploited in the wild, according to the Apache HTTP Server Project's core team's security advisory [1]. Tenable, the company behind the well-known vulnerability scanner Nessus, detected one hundred vulnerable web servers around the world at the time using Shodan.

After the patch was deployed, attackers continued to utilize it to gain access to sensitive information or run commands on web servers. Here are some significant instances:

- Posts on cybercrime forums, identified by CloudSEK's threat intelligence team, that described the vulnerability [5];
- Attacks against Asian high ranking officials, conducted by Chinese hackers [5];
- Mention in the Known Exploited Vulnerabilities Catalog of USA's Cybersecurity and Infrastructure Security Agency [8];
- Detections in Trend Micro's honeypots [7]; and
- Usage in Remote Access Tools and miners, for example by a Golang malware, H2Miner/Kinsing [6].

VI. Conclusions

In conclusion, despite the CVE-2021-41773 vulnerability's original severity, the white hat security community has demonstrated that an arbitrary file read can easily be translated into remote code execution capabilities. Despite the rapid release of a new version, courtesy of the Apache HTTP Server Project's staff, there were still unpatched servers on the Internet that cybercriminals were exploiting.

VII. Bibliography

[1] *Apache HTTP Server 2.4 vulnerabilities.* https://httpd.apache.org/security/vulnerabilities_24.html#CVE-2021-41773.

[2] *Apache HTTP Server CVE-2021-42013 and CVE-2021-41773 Exploited in the Wild.* https://blogs.juniper.net/en-us/threat-research/apache-http-server-cve-2021-42013-and-cve-2021-41773-exploited.

[3] *Apache HTTP Server Path Traversal Remote Code Execution (CVE-2021-41773 CVE-2021-42013).* https://blog.qualys.com/vulnerabilities-threat-research/2021/10/27/apache-http-server-path-traversal-remote-code-execution-cve-2021-41773-cve-2021-42013.

[4] *Apache HTTP Server Path Traversal Remote Code Execution (CVE-2021-41773 CVE-2021-42013).* ApacheHTTPServerCVE-2021-41773ExploitedintheWild.

[5] *Apache HTTP Server Project CVE-2021-41773 Vulnerability Actively Exploited in the Wild.* https://cloudsek.com/threatintelligence/cve-2021-41773-apache-http-server-exploited-actively-in-the-wild/.

[6] *CVE-2021-41773 Actively Exploited by H2Miner.* https://www.countercraftsec.com/blog/post/cve-2021-41773-actively-exploited-by-h2miner/.

[7] *How to detect Apache HTTP Server Exploitation.* https://www.trendmicro.com/en_my/devops/21/l/how-to-detect-apache-http-server-exploitation.html.

[8] *Known Exploited Vulnerabilities Catalog.* https://www.cisa.gov/known-exploited-vulnerabilities-catalog.

[9] *Suricata PT Open Ruleset.* https://github.com/ptresearch/AttackDetection.

[10] *Vulnerable docker images for CVE-2021-41773 Apache path traversal.* https://github.com/BlueTeamSteve/CVE-2021-41773.