PWNABLE.KR - collision

Lets connect to our server,

```
ra@moni~> ssh col@128.61.240.205 -p 2222
col@128.61.240.205's password:
 _____  __    __  _____    _____  _____   _        ___      __  _  _____
|    \|  |__|  ||    \  /    ||    \ | |      /  _]    |  |/ ]|    \
|  o  )  |  |  ||  _  ||  o  ||  o  )| |     /  [_     |  ' / |  D  )
|   _/|  |  |  ||  |  ||     ||     || |___ |    _]    |    \ |    /
|  |  |  `  '  ||  |  ||  _  ||  O  ||     ||   [_  __ |     \|    \
|  |   \      / |  |  ||  |  ||     ||     ||     ||  ||  .  ||  .  \
|__|    \_/\_/  |__|__||__|__||_____||_____||_____||__||__|\_||__|\_|

- Site admin : daehee87@gatech.edu
- IRC : irc.netgarage.org:6667 / #pwnable.kr
- Simply type "irssi" command to join IRC now
- files under /tmp can be erased anytime. make your directory under /tmp
- to use peda, issue `source /usr/share/peda/peda.py` in gdb terminal
You have new mail.
Last login: Wed Jun  2 03:52:15 2021 from 49.205.143.178
col@pwnable:~$
```

After listing the files in it, we can see

```
col@pwnable:~$ ls -la
total 36
drwxr-x---   5 root    col     4096 Oct 23  2016 .
drwxr-xr-x 115 root    root    4096 Dec 22 08:10 ..
d---------   2 root    root    4096 Jun 12  2014 .bash_history
-r-sr-x---   1 col_pwn col     7341 Jun 11  2014 col
-rw-r--r--   1 root    root     555 Jun 12  2014 col.c
-r--r-----   1 col_pwn col_pwn   52 Jun 11  2014 flag
dr-xr-xr-x   2 root    root    4096 Aug 20  2014 .irssi
drwxr-xr-x   2 root    root    4096 Oct 23  2016 .pwntools-cache
col@pwnable:~$
```

It seems like there are some restrictions in privilege as usual,

```
col@pwnable:~$ cat flag
cat: flag: Permission denied
col@pwnable:~$
```

We cannot read our flag directly, so we have to use the binary to get it

By analyzing the file type of our binary using `file` command, we get

```
col@pwnable:~$ file col
col: setuid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked, interpreter /lib/ld-, for GNU/Linux 2.6.24,
BuildID[sha1]=05a10e253161f02d8e6553d95018bc82c7b531fe, not stripped
```

Now lets view the source code of the binary from `col.c`

```
col@pwnable:~$ cat col.c
#include <stdio.h>
#include <string.h>
unsigned long hashcode = 0x21DD09EC;
unsigned long check_password(const char* p){
    int* ip = (int*)p;
    int i;
    int res=0;
    for(i=0; i<5; i++){
        res += ip[i];
    }
    return res;
}

int main(int argc, char* argv[]){
    if(argc<2){
        printf("usage : %s [passcode]\n", argv[0]);
        return 0;
    }
    if(strlen(argv[1]) != 20){
        printf("passcode length should be 20 bytes\n");
        return 0;
    }

    if(hashcode == check_password( argv[1] )){
        system("/bin/cat flag");
        return 0;
    }
    else
        printf("wrong passcode.\n");
    return 0;
}
```

Now after analyzing the binary, we can come to a conclusion that,

- It has a global variable `unsigned long hashcode = 0x21DD09EC;`
- It has a global function `unsigned long check_password()`
- It gets two arguments (ie. filename,input1)

- It compare the input length which is equal to 20 bytes or not
- If it passes these conditions, it compares the `hashcode` with `check_password(input1)`
- If it is true, it displays the `flag`
- Else it displays an error message

So the whole binary lies on this function,

```
unsigned long check_password(const char* p){
    int* ip = (int*)p;
    int i;
    int res=0;
    for(i=0; i<5; i++){
        res += ip[i];
    }
    return res;
}
```

The value `res` being returned from this function should be equal to `0x21DD09EC`

We knew the resultant value should be `0x21DD09EC`,

The value in `int` is,

```
>>> print(0x21DD09EC)
568134124
```

It is being looped 5 times and added to `res`,so

```
>>> print(0x21DD09EC/5)
113626824.8
```

The function `check_password(const char* p)` gets its argument in `char`

But it typecasts itself into `int` array by `int* ip = (int*)p;`

Actually we need it to pass as `char` in input,not `int`

Maximum size of `int` is `4 bytes`

So the `int` array would be split into `4 bytes` for each loop,thus `( 5 loop * 4 bytes = 20 bytes )`

Now,lets guess the input

`res` = `0x21DD09EC` = `568134124`

it should be in 5 parts (a,b,c,d,e)

```
>>> print(0x21DD09EC/4)
142033531.0
>>> print(0x21DD09EC%4)
0
>>> print(0x21DD09EC/5)
113626824.8
>>> print(0x21DD09EC%5)
4
```

So it can have 4 parts of equal value, if we go for 5 it gives float which cannot be converted into `hex` (only `int` to `hex` possible)

Converting it into `hex`

```
>>> hex(142033531)
'0x877427b'
```

The total value of 4 equal parts is equal to our needed `res`

```
>>> hex(142033531*4)
'0x21dd09ec'
```

But we cannot pass `\x00` null bytes into our input, because it will get terminated and our program won't pass

So lets find the other way, by sharing values with 5 parts

```
>>> print(0x21DD09EC/5)
113626824.8
>>> print(0x21DD09EC%5)
4
>>> print(0x21DD09EC//5)
113626824
>>> print(568134124-(113626824*4))
113626828
>>> hex(113626824)
'0x6c5cec8'
>>> hex(113626828)
'0x6c5cecc'
```

So our first 4 parts have `0x6c5cec8` and the last part as `0x6c5cecc`

Lets craft our string with these to pass as input,

```
ra@moni~> python3 -c 'print("\xc8\xce\xc5\x06" * 4 + "\xcc\xce\xc5\x06")'
ÈÎÅÈÎÅÈÎÅÈÎÅÌÎÅ
```

Its time for our exploit,

We will be using `pwntools`

To install it pass `pip install pwntools` in yout terminal

Code for our exploit,

```
ra@moni~/P/pwnable.kr> cat col-exploit.py
from pwn import *
data=p32(0x6c5cec8)*4+p32(0x6c5cecc)
remote = ssh('col' ,'128.61.240.205' ,password='guest', port=2222)
program = remote.process(executable='./col', argv=['col',data])
flag = program.recv()
log.success("Flag: " + flag)
program.close()
remote.close()
```

Lets try our final exploit

```
ra@moni~/P/pwnable.kr> python col-exploit.py
[+] Connecting to 128.61.240.205 on port 2222: Done
[*] col@128.61.240.205:
    Distro    Ubuntu 16.04
    OS:       linux
    Arch:     amd64
    Version:  4.4.179
    ASLR:     Enabled
[+] Starting remote process u'./col' on 128.61.240.205: pid 428418
[+] Flag: daddy! I just managed to create a hash collision :)
[*] Stopped remote process u'col' on 128.61.240.205 (pid 428418)
[*] Closed connection to '128.61.240.205'
```

Done! Our flag is,

`Flag: daddy! I just managed to create a hash collision :)`