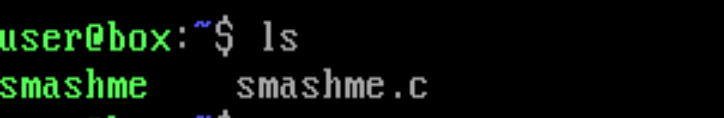
Rupeng Na 250884549

Assignment4 CTF Challenge

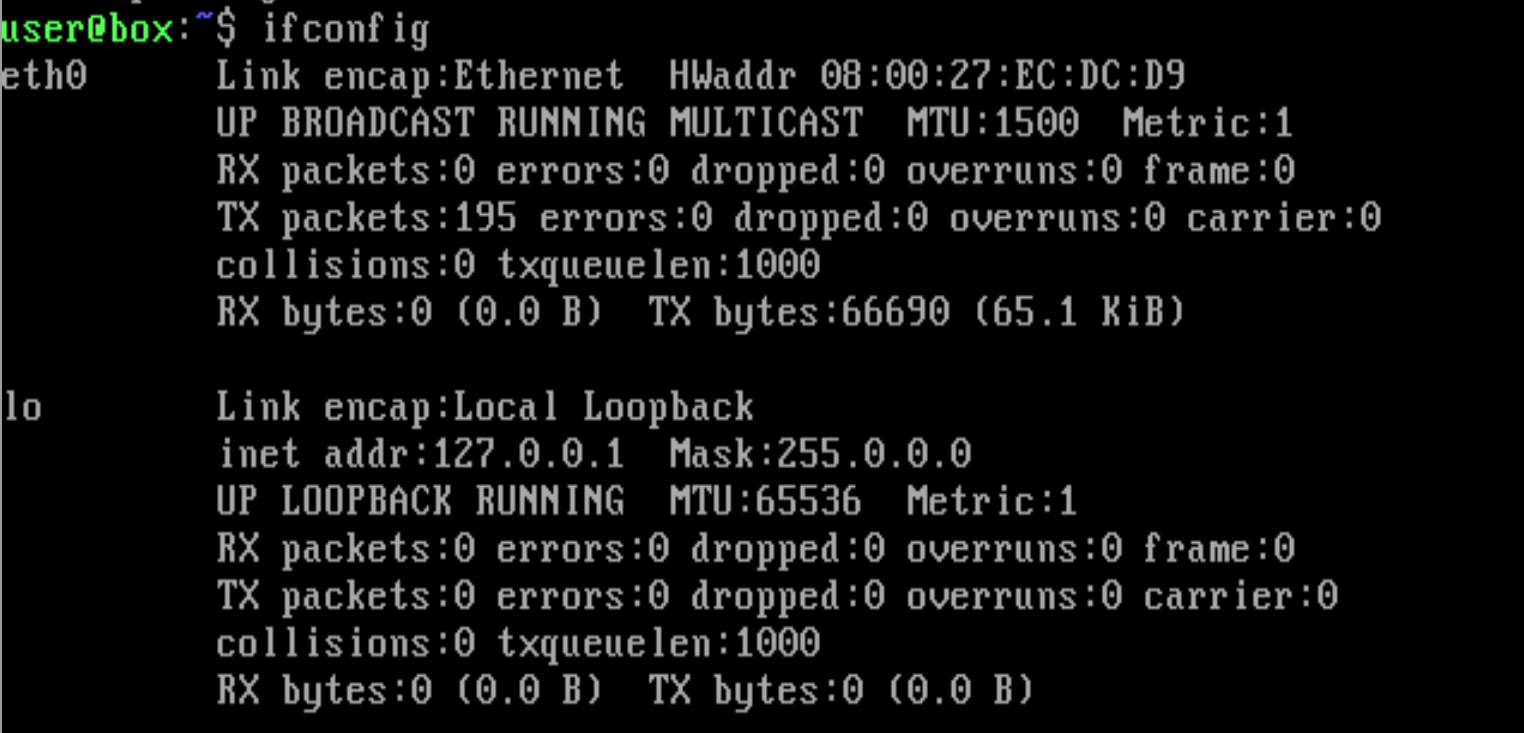
At first I logged into the VM. The account and password are user.



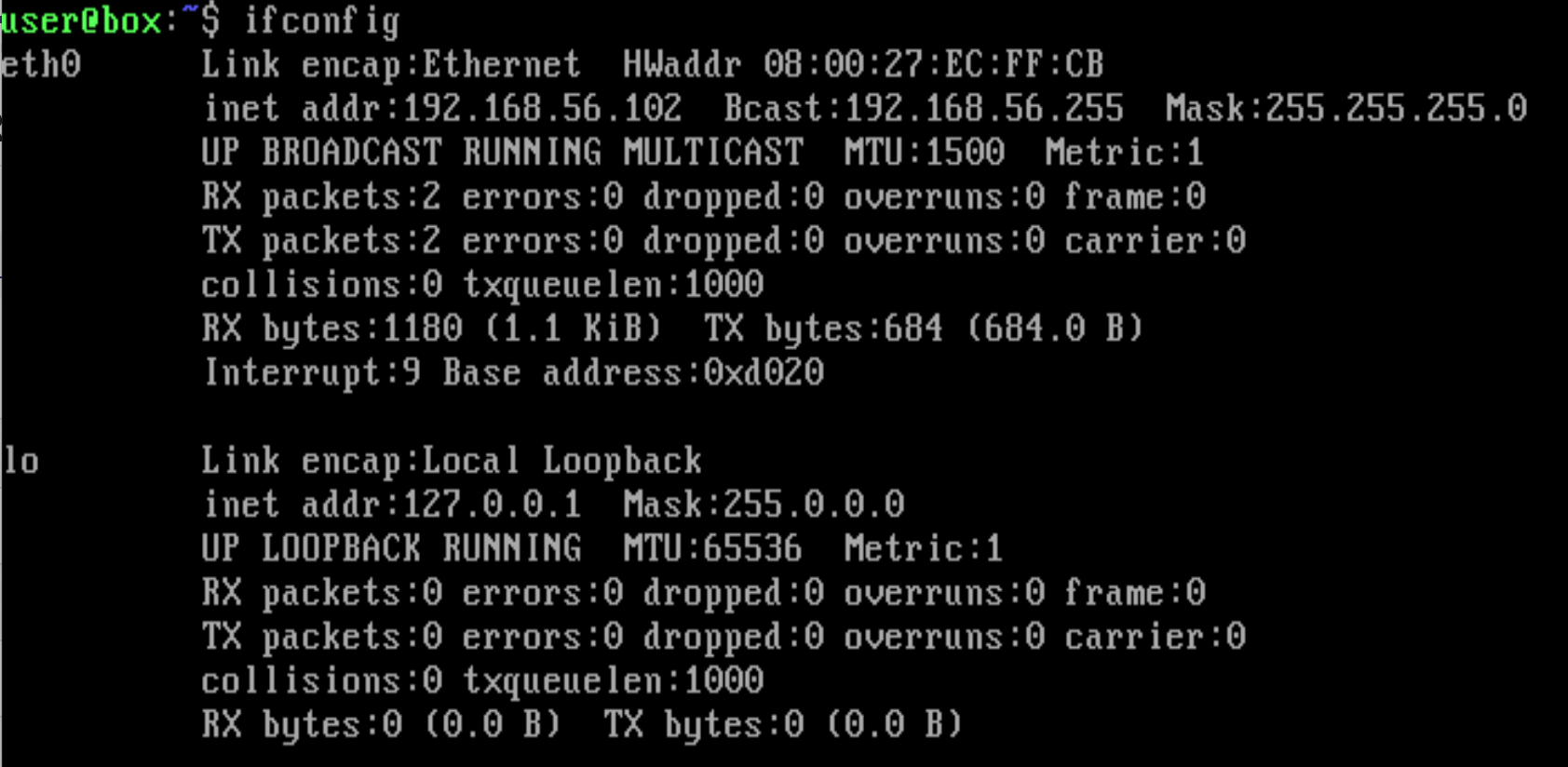
I always use ls to see the directory after I logged into the VM. There’s only two files, one of them are C file and another one is an object file.

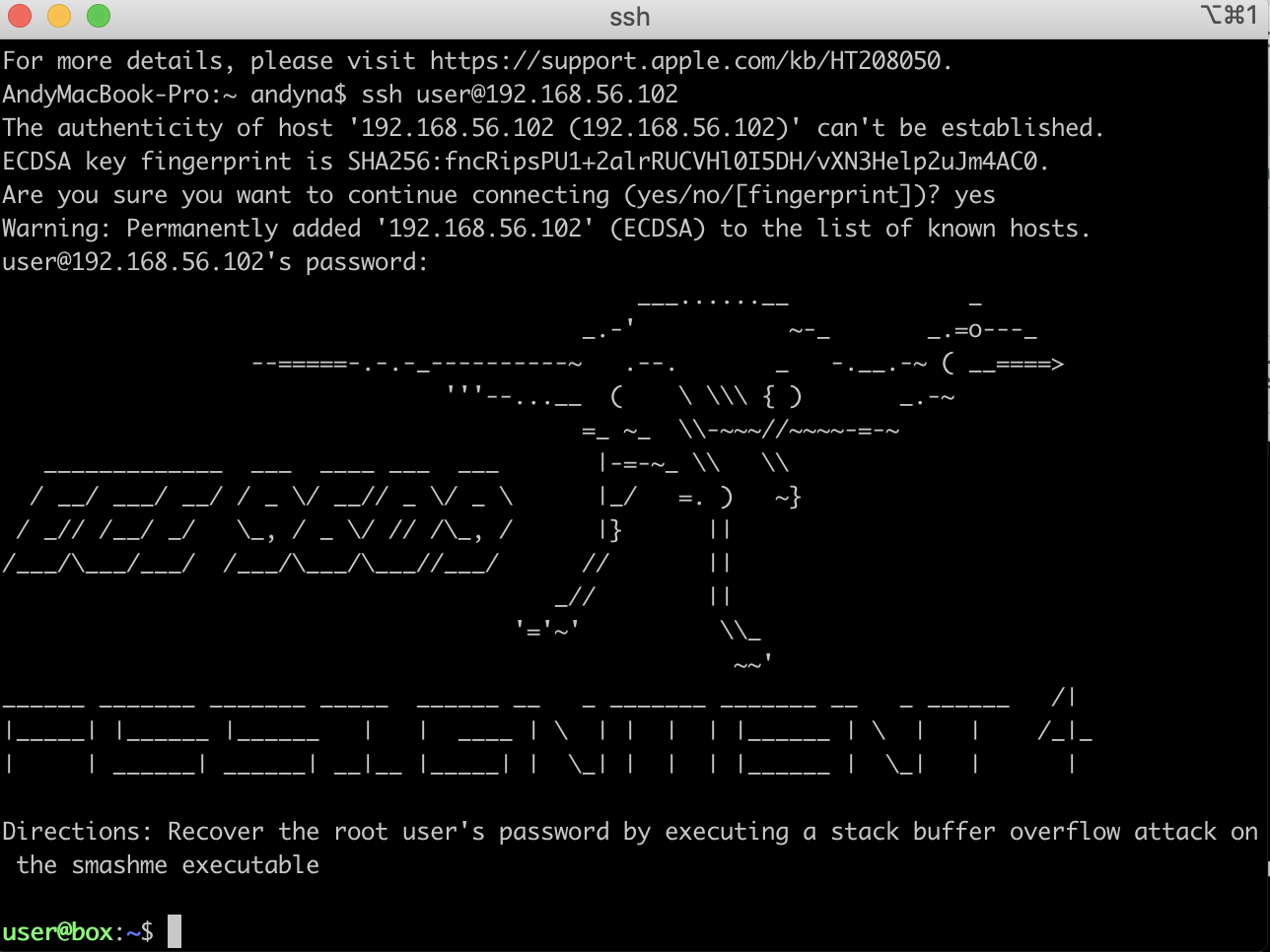


Similar to assignment 3, I need to use ssh instead of directly in the VM because it is easier to do things like copy and paste. I need to get the IP address by ifconfig. When I use Windows and set Network adapter type to paravirtualized Network, I cannot find the inet address at eth0. It is wired, but I think there is something wrong with the network adapter.

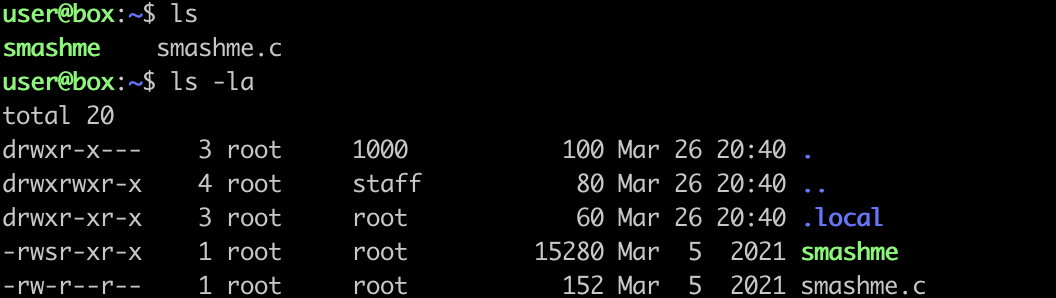


Then I try to do it again with my Mac. It was successful to get the inet address, which is 192.168.56.102.

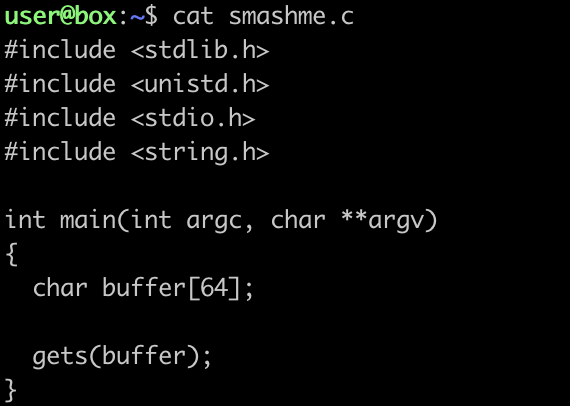


I use terminal to ssh to the VM:

List all files in the directory by ls -la



I opened smashme.c to see how it works. It is the same with stack 5 of Protostar. As the video said, get() function is a dangerous function. It will break the security because it needs to know how many characters the function already reads and it will still store characters past the end of the buffer.



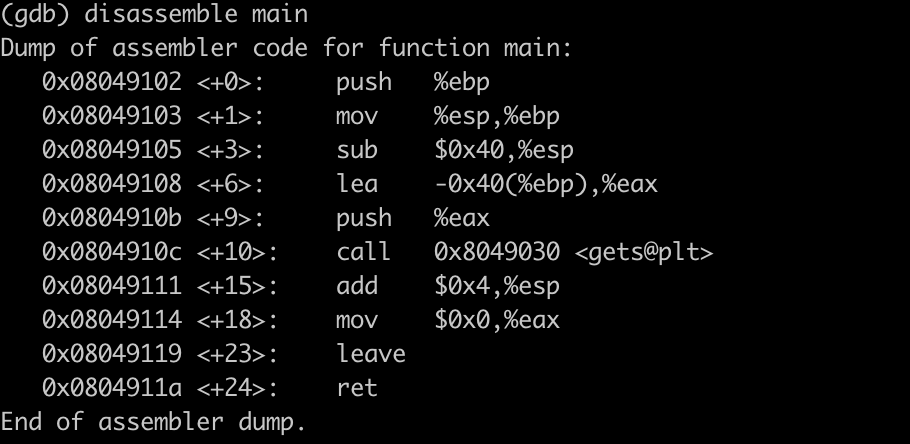
As the instruction of assignment 4 shows, there are two steps to get the access to the /etc/shadow and recover the root user’s password:

1.Phases of stack buffer overflow exploit;

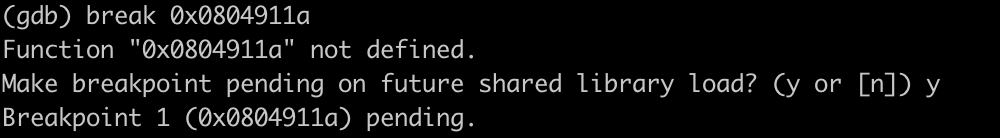
2.Password Cracking

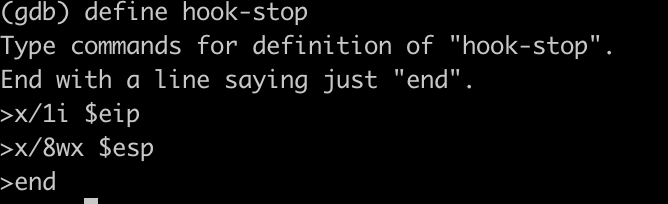
**Phases of stack buffer overflow exploit**

As the tutorial said, we need to find a way to get from no functionality of the program to the root shell.

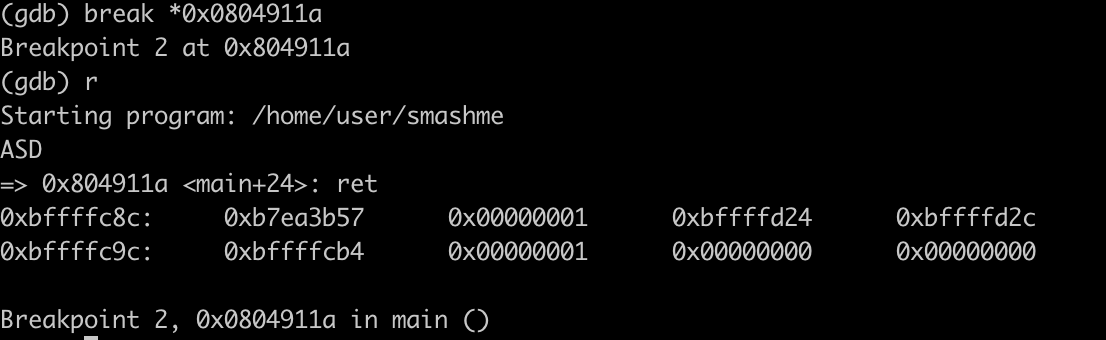


I followed the tutorial, set the break at the return of main. I made a mistake here, everytime set the breakpoint, we need to add ‘\*’ before the address.

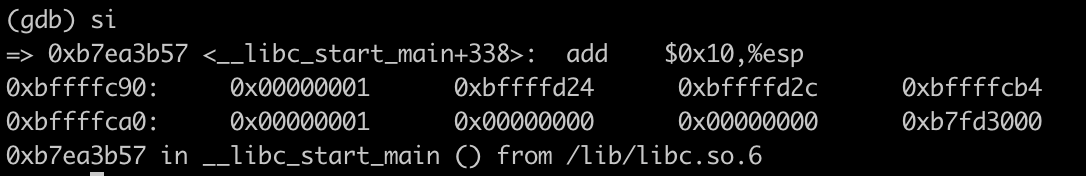
Hook-stop is a special definition that GDB calls at every breakpoint event. This means you can use it to call user-defined functions every time GDB stops.

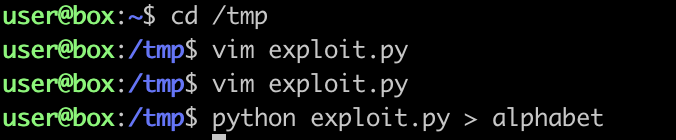


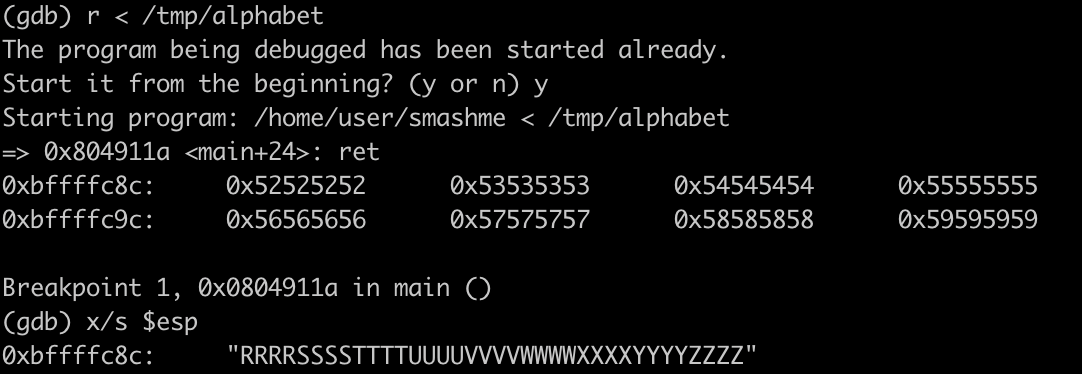
Then I run the program without any excution. It show the memory in the 8c and 9c.



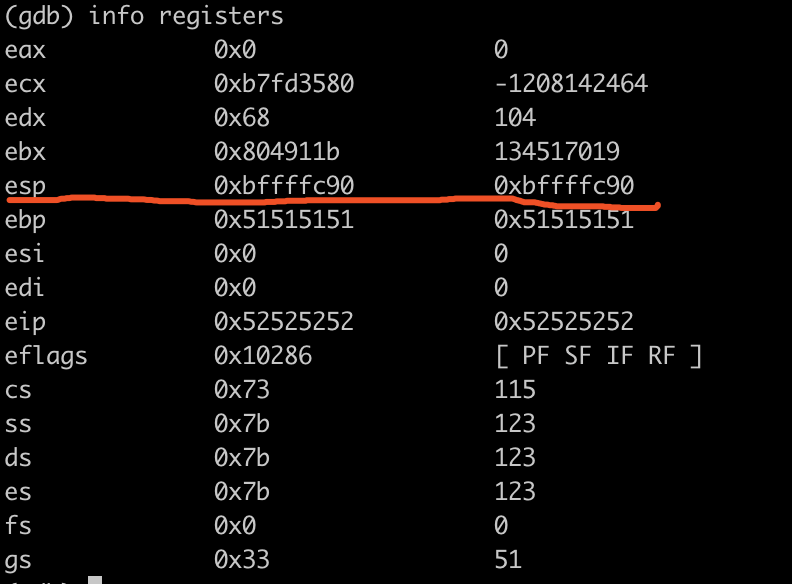
Execute a line of source code, if there is a function call in this line of code, enter the function.

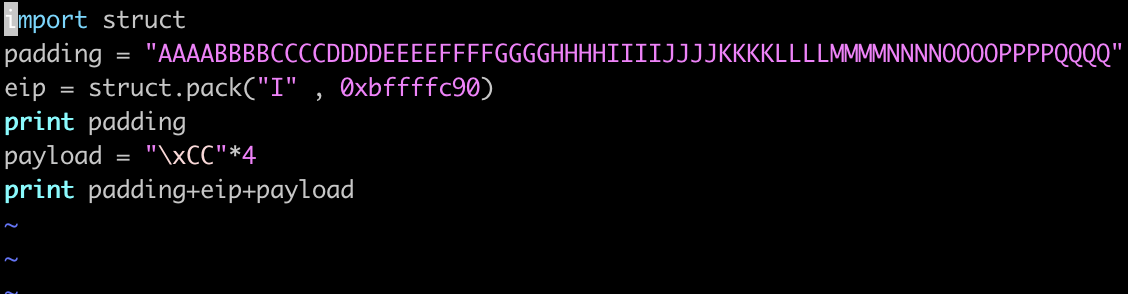


Here, I use another terminal to write a simple pattern of letter for the overflow. Then script it into a file ‘alphabet’

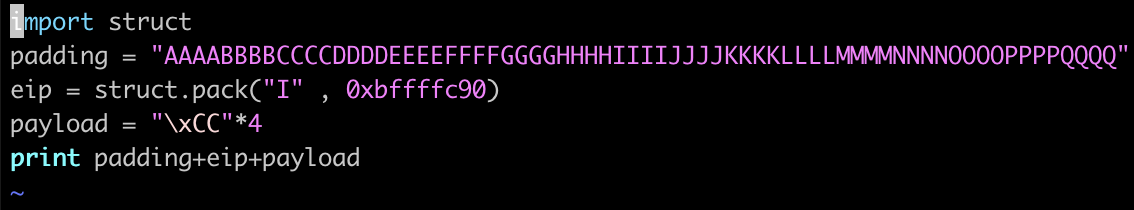
Then put the file into the program. When the content gets into the program, it may cause overflow. Our target is to figure out the overflow part. As we mentioned, the 8c and 9c parts changed to the alphabet which is started by ‘RRRR…” It shows that overflow will happen after the ‘Q’ and before parts will store in memory. The esp proves that.

The esp means the stack pointer. The info register is used to find the address I will use to point to the program execution. Here, I will give the instruction pointer an address where the execution overflows the memory. The stack pointer always points to the last item put on the stack; the overflowed memory will follow: 0xbffffc90.

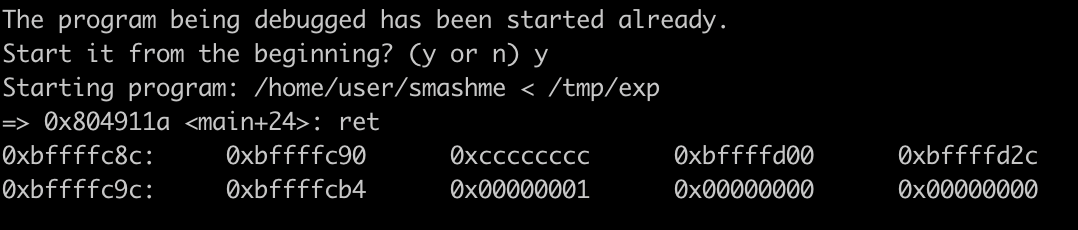


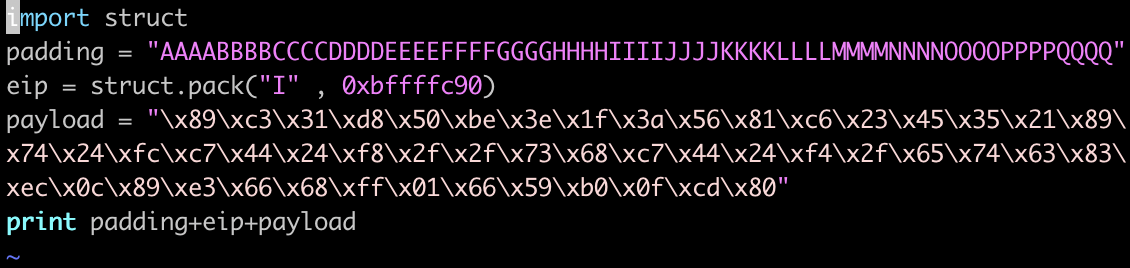
A similar exploit code is shown in the tutorial. eip is the instruction pointer, and its esp address is 0xbffffc90. struct.pack("I", address) is used to pack the long integer into the parameter eip, "I" represents the data type of the long integer. eip will get the address of the end of the stack and execute instructions from the end of the stack upwards. 

exp is used to save the result of exploit.py. Note, we need to modify the exploit.py before we execute it.

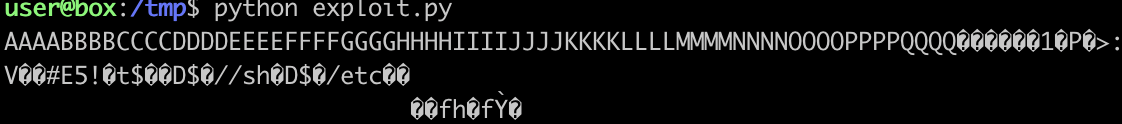


Then put the exp back to the program and run the smashme again. The stack has already changed to 0xbffffc90. Note the 0xccccccc is the payload we set up in the exploit.py which is the interrupt parts. When the 0xcccccccc occurred, it means we successfully interrupted.



Next, I change the payload by the shell code to payload only. 

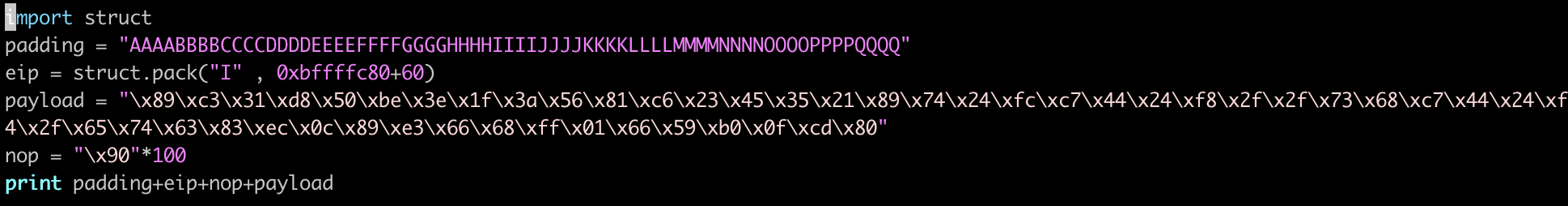
The output is seems like:



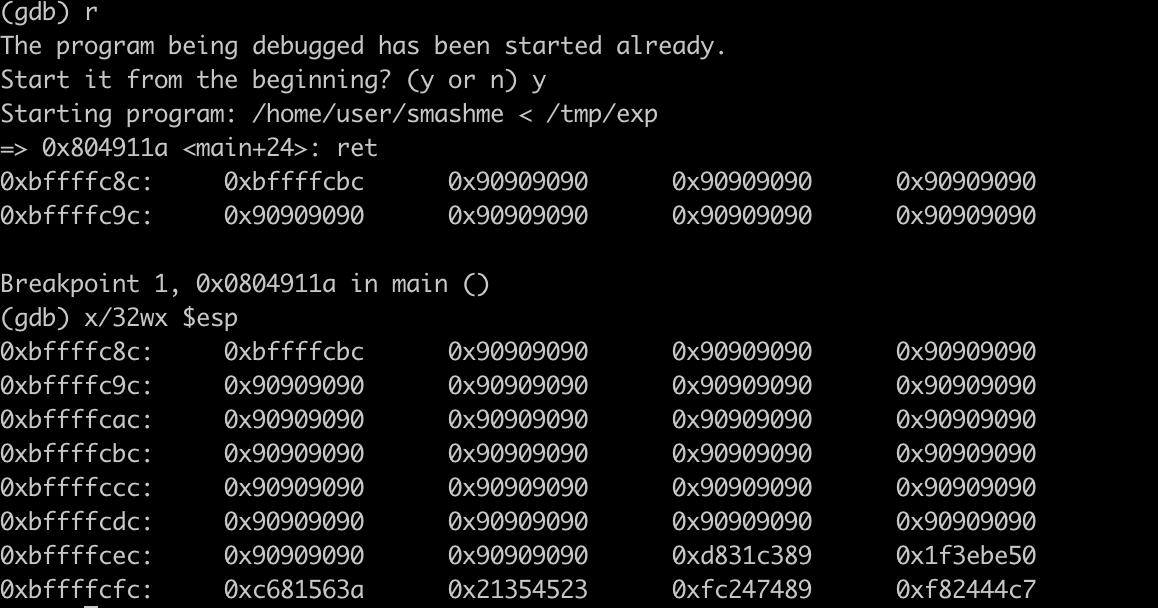
When I try to put it into smashme, I got the segmentation fault. Then I went back to watch the video again. I found I forgot to set nop to stop the changing of esp.



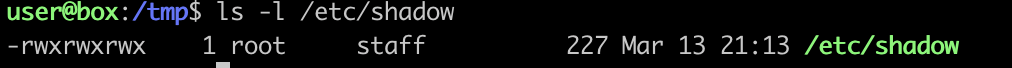
I set the nop for the same environment, the nop is 100 times, and the offset is 60 times. Note that, at first I set the offset to 30, the esp didn’t contain the nop slides(0x90909090).



After we run it again, we found the nop(0x90909090) already fulfill the register



At last, I go to check the shadow file’s permission. It has already changed to 777.



**Password Cracking:**

First time I saw the following content, I didn't understand it. So I went to search for the definition of the shadow file structure. <https://www.cyberciti.biz/faq/understanding-etcshadow-file/>. I found that the part before the ‘:’ is the username which means the login account. The part after the

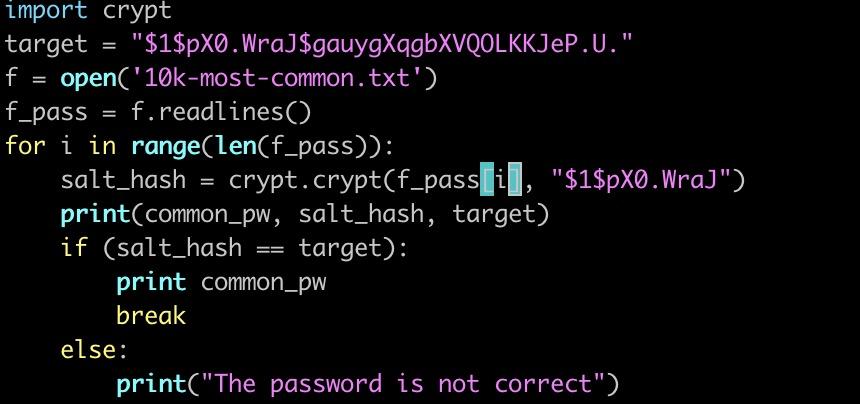
‘:’ is the encrypted password. As the tutorial showed, password format is set to **$id$salt$hashed**. The $id indicates which type of encrypt method we use. In our shadow file, the $id is $1, which means we used MD5.

The instruction of the assignment gives us a file that contains 10000 common passwords. So I want to encrypt each of them by MD5 then compare them with the root’s password. If there’s a word’s equals the password, then we can enter the system as the root.

First, we need to build the 10000 passwords file. Only one thing we need to mention, our vm did not connect to the Internet. We can not use wget to download directly from the Internet. I just copy the file and paste to the 10k-most-common.txt.



I write a file to compare the root’s password with each of the words after encrypting. At first we need to read each of the lines as the word we need. Then encrypt it with the same $id$salt way. At last I try to compare each of the words in the file with the root’s password.



Then use ‘python password\_cracking.py’ to execute the program. The program seems to pass. It show lots of fail until find the root password which is ‘idunno’.



Before we test the password, we need to change the user from ‘user’ to the root. The command to approve that is: ‘su’. Then I enter the password ‘idunno’. It worked!

