# 实验二 缓冲区溢出漏洞分析与验证

班级：1703202

学号：1170300421

专业：信息安全

日期：2020.10.25

# 实验二 缓冲区溢出漏洞分析与验证

## 1、实验需知

1. 指导书中横线范围内为终端运行结果
2. 带下划线的字段为输入的命令
3. 虚拟机中释放鼠标的快捷键为Ctrl+Alt

## 2、实验预备

本节课中熟悉实验环境，分析一个Web服务器的逻辑，寻找缓冲区溢出漏洞并触发该漏洞。

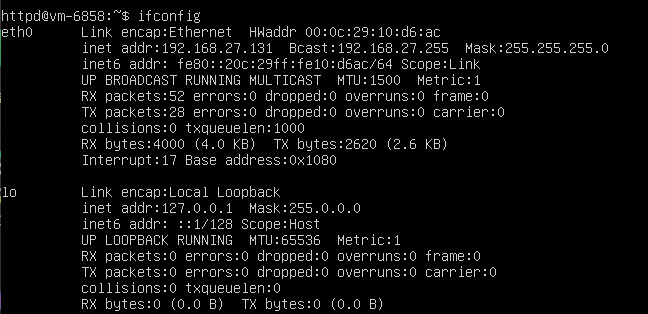
实验环境为Ubuntu，在VMware Player 虚拟机中的vm-6858运行。系统中有两个账号：

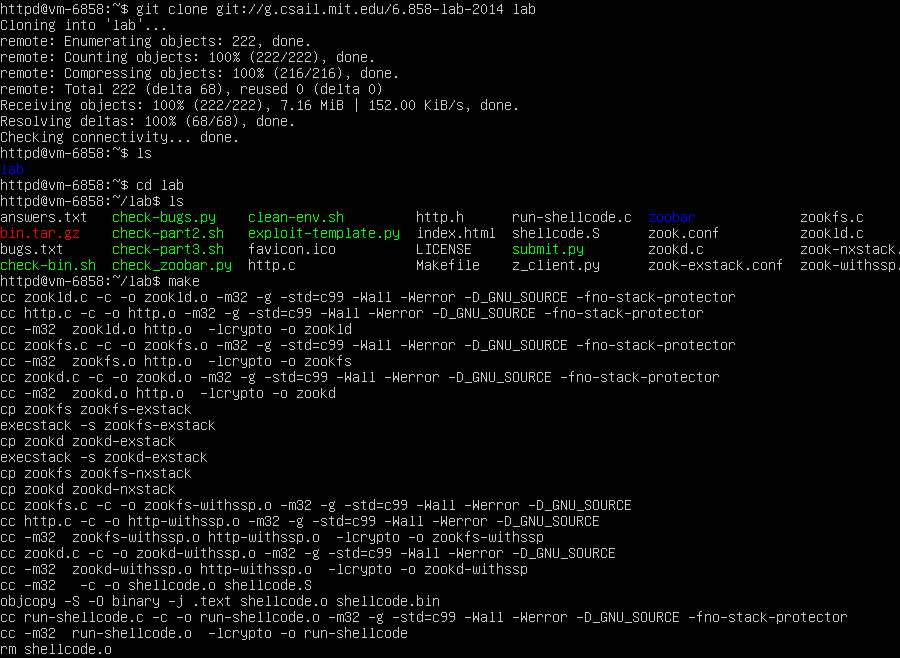
- `root`，口令6858，用来安装软件  
- `httpd`，口令6858，运行Web服务器和实验程序

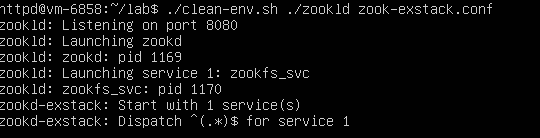
本课程实验研究对象是一个web服务器`zookws`。该服务器上运行一个Python的web应用`zoobar`，web用户之间转移一种称为“zoobars”的货币。

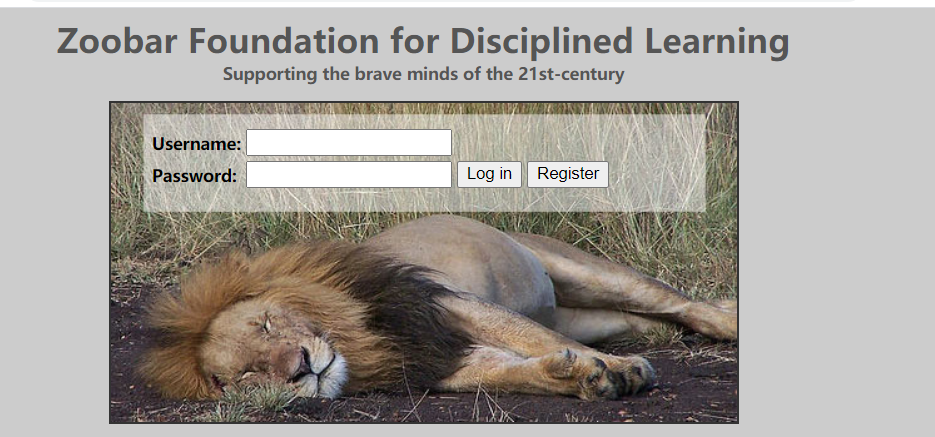
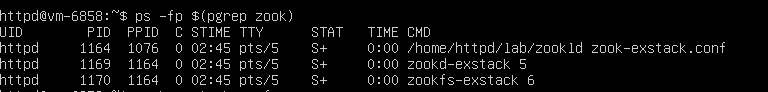
1. 在VMware里用`httpd`账号登录后，运行`ifconfig`查看IP地址



  
 2. 用终端软件Xshell通过SSH登录系统`ssh httpd@IP地址`  
 3. 在/home/httpd/lab路径下`make`编译程序

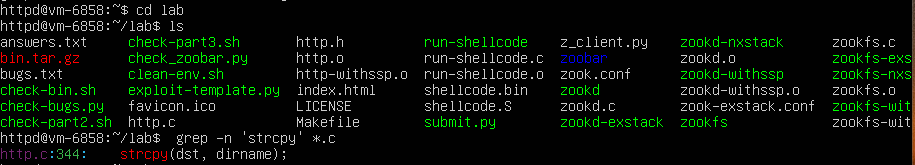
  
 4. 启动服务器`./clean-env.sh ./zookld zook-exstack.conf`

  
 5. 用浏览器访问zook服务`http://虚拟机IP地址:8080/`

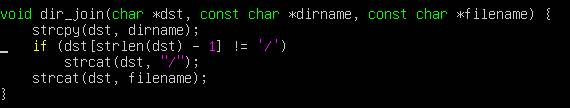
  
查看一下被启动的程序：  


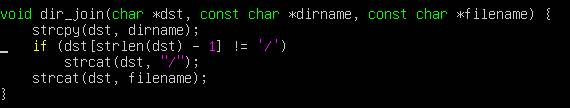
## 3、寻找漏洞

缓冲区溢出存在的要素：数组（字符串），串处理/读取函数（写操作）。  
数组：`char \* s`, `char s[128]`, `int a[128]`, `void \* p`。  
函数： `strcpy()`, `strcat()`, `sprintf()`, `vsprintf()`, `gets()`, `getc()`, `read()`, `scanf()`, `getenv()`。  
除了调用函数外，还可能通过`for/while {}`循环的方式来访问缓冲区。  
1、在源码中搜索一个‘危险’函数`strcpy()`。

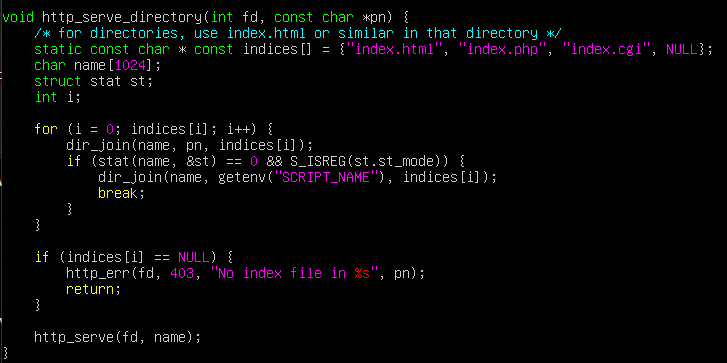


在`http.c`中找到了一处潜在漏洞，来具体看一下代码。（line 343）

   
`dir\_join()`函数将`dirname`和`filename`先后拷贝到`dst`中。显然，这里并没有检查每一个字符串长度。若`dirname`长度比`dst`缓冲长，则`strcpy()`调用存在缓冲区溢出风险。  
  
2、进一步检查使用`dir\_join()`时是否存在导致缓冲区溢出的可能（line 350）：



在`dir\_join(name, pn, indices[i]);`调用中，`char name[]`长度为1024。`char \* pn`长度待定。

1. 继续查看`http\_serve\_directory()`调用情况。（line 273）  
   

handler = http\_serve\_directory`，`handler()`中的`pn`长度1024，内容来自`getcwd()`加上`strcat(pn, name)`。若`name`过长，则`pn`长度也将过长。

1. 通过进一步分析`http\_serve()`调用过程，发现`name`内容来自于环境变量`REQUEST\_URI`。  
   zookfs.c:47:

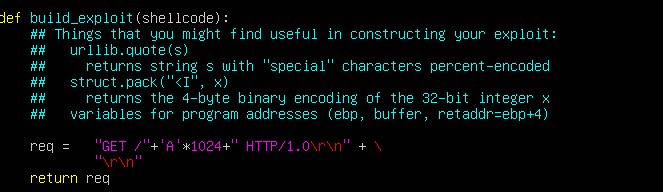
该环境变量在`http.c`中`http\_request\_line()`函数中被设置。  
http.c:107:   
  
在`zookd.c`中，`http\_request\_line()`函数被`process\_client()`函数调用。  
zookd.c:70:

  
我们把这一漏洞命名为“LONG\_URI”漏洞

## **4、触发漏洞**

首先，该漏洞必须能改写栈中的一个返回地址；其次，改写一些数据结构来用于夺取程序的控制流。撰写触发该漏洞的程序，并验证改程序可以导致web服务器崩溃（通过`dmesg | tail`, 使用`gdb`, 或直接观察）。  
漏洞利用程序模板为`exploit-template.py`，该程序向服务器发送特殊请求。  
1、首先启动服务：`./clean-env.sh ./zookld zook-exstack.conf`。  
打开另一终端，执行`exploit-template.py`。下面是是未改写的`exploit-template.py`执行结果。

注：返回的内容不一定为success，只需保证HTTP response内容不为空即可。

我们发现了可由我们控制的用户输入，客户端请求位于下面代码中：   


目前，我们手上有了两个攻击服务器的武器：(1) “LONG\_URI”缓冲区溢出漏洞，(2)构造请求输入`req`的脚本。下一步就是要分析`http.c`中处理该请求的代码，将`req`中内容和`REQUEST\_URI`对应起来。  
  
通过分析代码可以发现，HTTP请求中的路径，例如`/foo.html`，被赋予了`REQUEST\_URI`变量，因此可以通过构造较长的HTTP请求路径来令缓冲区溢出。  
2、之前发现缓冲区有1024字节，我们就令请求路径超过1024字节。  
具体修改方法如下：

2.1 输入`vim exploit-template.py`进入文件开始修改

httpd@vm-6858:~/lab$ vim exploit-template.py

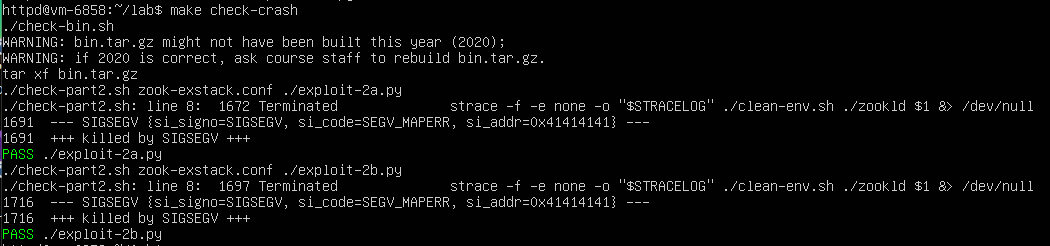
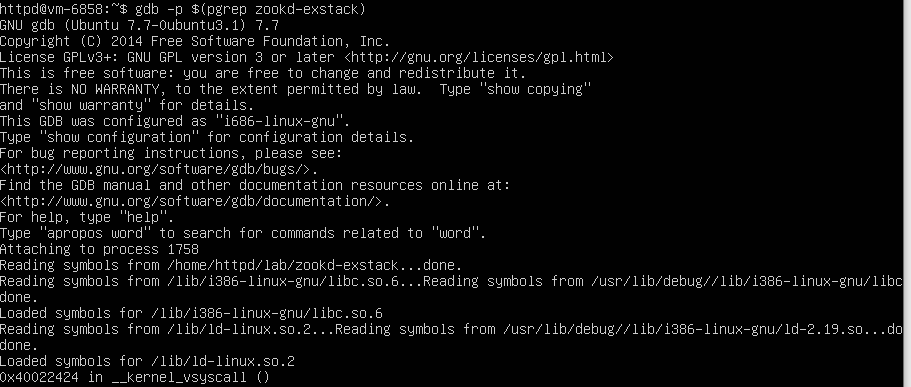
2.2 按下字母`i`，进入插入模式修改文件内容，令请求路径超过1024字节

req = "GET /" + 'A' \* 1024 + " HTTP/1.0\r\n" + \  
 "\r\n"

2.3 修改完成后，按下“Esc”键转到命令行模式

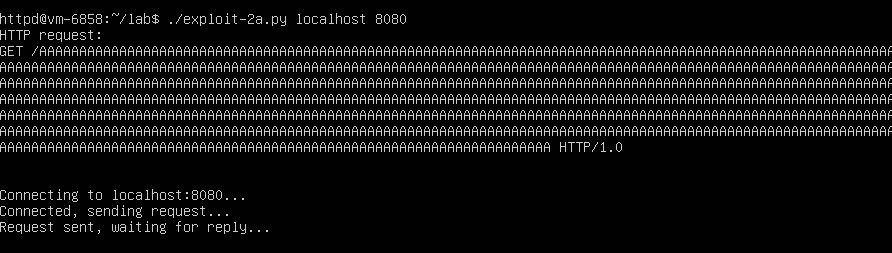
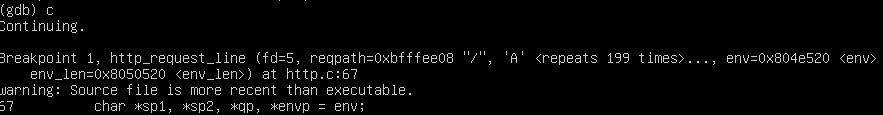
2.4 将攻击脚本复制两份，命名为`exploit-2a.py`和`exploit-2b.py`，并修改文件的权限，使其可执行。

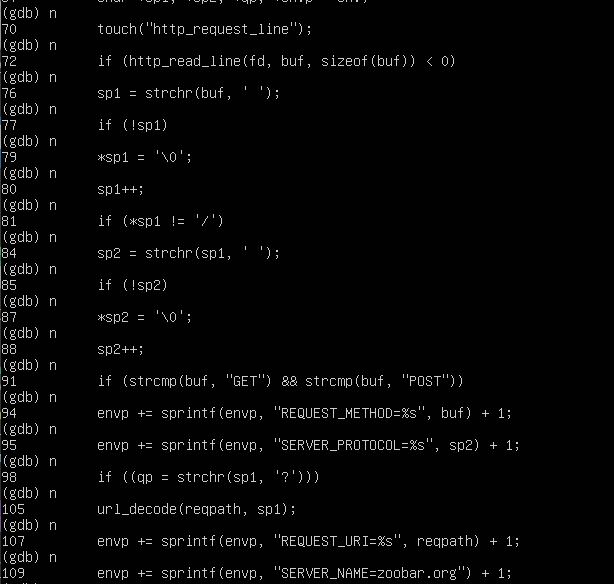
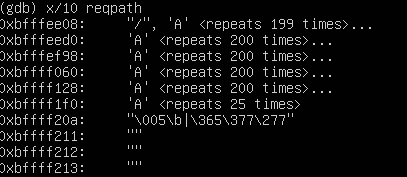
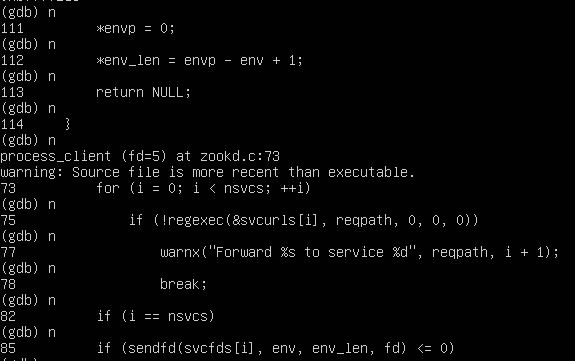


3、用`make check-crash`来验证是否导致程序崩溃。  
该程序并通过(PASS)了检查。缓冲区漏洞导致程序因为SIGSEV信号而崩溃，指令地址被改写为`si\_addr=0x41414141`。下面看看具体发生了什么。  
4.1 首先，调试`zookd`。`http\_request\_line`负责处理HTTP请求。  


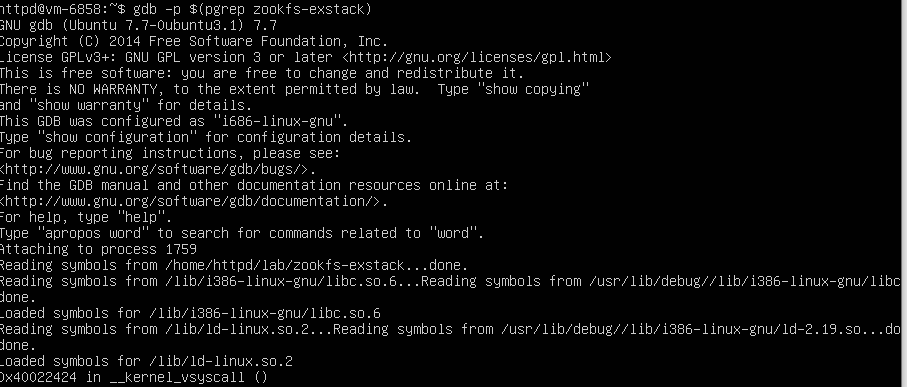


[运行漏洞触发程序]

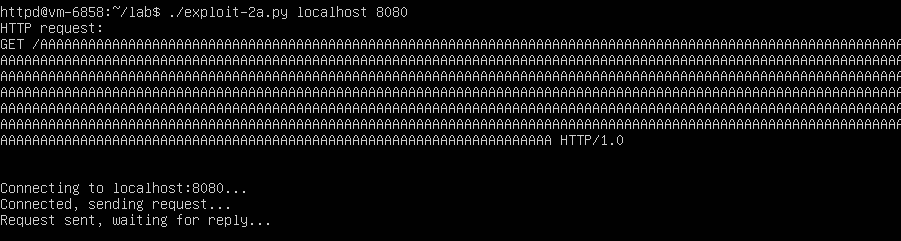
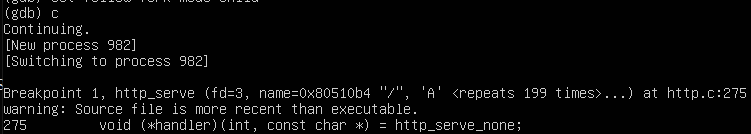
  
 (gdb) n

  
[执行n多次直到REQUEST\_URI被处理完]  
  
"  
(gdb) x/10 reqpath [打印reqpath，为"/", A \* 1024]  
  
[继续用n命令执行]  
  
[此时zookd将请求发送给zookfs]

(gdb) p sizeof reqpath [reqpath不会溢出]  
  
(gdb) p sizeof env [env不会溢出]  
  
(gdb) quit  
 `zookd`此时并不存在缓冲区溢出。

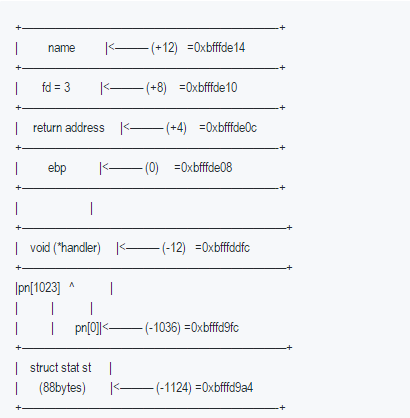
4.2接下来分析`zookfs`。  
在`zookfs.c`中，`http\_serve()`函数以`REQUEST\_URI`环境变量为参数。在`http\_serve`处设置断点，分析栈结构。  


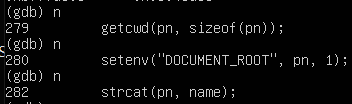
  
[运行漏洞触发程序]

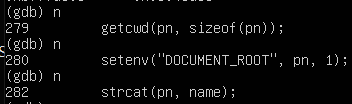
(gdb) c  
(gdb) p $ebp



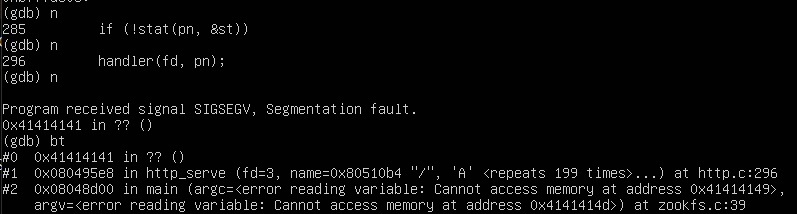
(gdb) p &handler  
  
(gdb) p &pn  
  
(gdb) p &st  
  
(gdb) p &fd  
  
(gdb) P &name  
  
(gdb) x $ebp+4  


根据上面的调试信息绘制`http\_serve()`的栈结构：  


继续执行到`strcat()`，  


  
(gdb) p pn  
  
(gdb) p strlen(name)  
  
(gdb) p sizeof pn  
  
此处将执行`strcat()`，在`pn`中已经包含的来自`getcwd()`的字符串后面加上长度1025的`name`，将超过`pn`所分配的大小1024，导致缓冲区溢出。接着执行一步，并查看缓冲区溢出情况。

 ""

在`pn`之前的缓冲区，包括`handler`和`$ebp`，已经被字符`A`覆盖，但返回地址并没有被完全改写。  


继续执行到`handler(fd, pn);`，由于`handler`变量被改写为`0x41414141`，导致程序崩溃。这发生在先前发现的`strcpy()`漏洞之前。攻击成功。