**信息安全实验报告**

**Lab 3 Shellshock**

**孙铁**

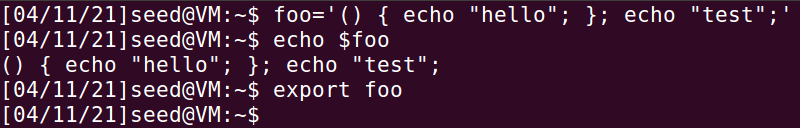
**SA20225414**

**Task 1**

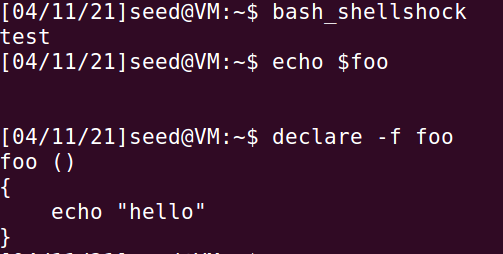
实验所使用的Ubuntu 16.04系统有两种Bash程序：Bash和Bash\_Shellshock。区别在于Bash\_Shellshock没有针对Shellshock攻击的补丁，无法防御此种攻击。

通过实验进行验证：

Bash：



Bash\_Shellshock：



可以看到在Bash\_Shellshock中，foo变量第二个分号后面的部分被执行。

**Task 2**

准备两台虚拟机：

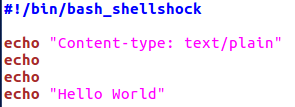
攻击者10.0.2.7：



目标服务器10.0.2.8



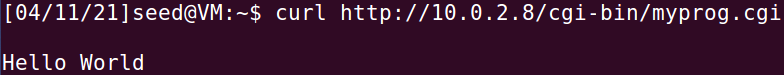
编写CGI程序myprog.cgi放在目标服务器(10.0.2.8) 中 /usr/lib/cgi-bin路径下，其作用是打印出“Hello World”。



将文件权限设置为755 (令其可执行)

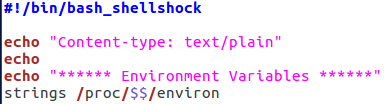


在攻击者主机(10.0.2.7)上，使用curl命令，结合目标主机IP地址，进行攻击：



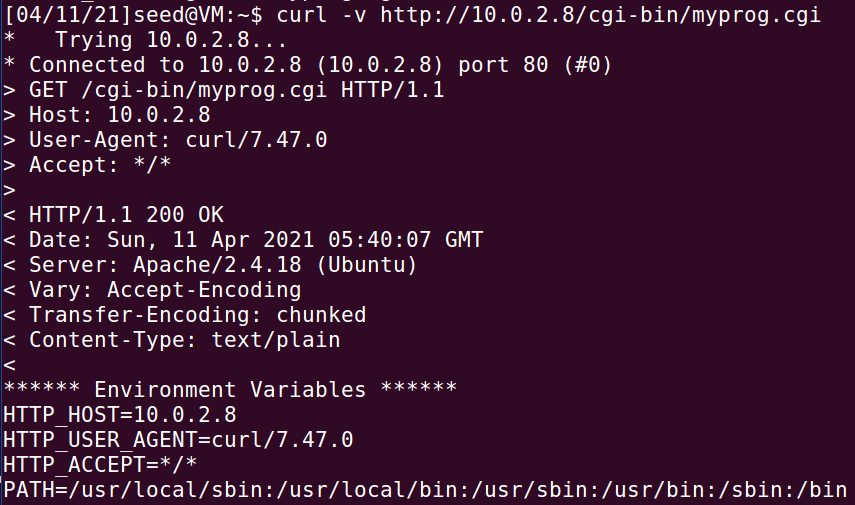
**Task 3**

修改目标服务器(10.0.2.8)中 /usr/lib/cgi-bin路径下的CGI程序myprog.cgi，其作用是打印出环境变量。



文件权限设置为755。

在攻击者主机(10.0.2.7)上，使用curl命令 -v 选项打印出HTTP请求和服务器的响应：



尝试从攻击者主机(10.0.2.7)向目标服务器(10.0.2.8)发送数据，curl命令的一些选项可以用来设置请求字段，根据上图可以看出，User-Agent字段的信息与目标服务器环境变量HTTP\_USER\_AGENT完全相同，说明服务器从HTTP请求中获取User-Agent字段，并将其赋值给环境变量HTTP\_USER\_AGENT。

实验进行验证：

使用curl -A 命令修改HTTP请求的User-Agent字段为“test”：



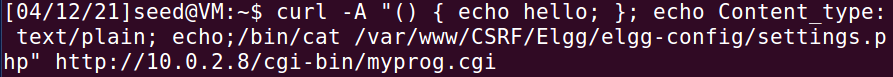
目标服务器(10.0.2.8)中的环境变量果然被修改。

因为CGI程序myprog.cgi开头是“#！/bin/bash\_shellshock”，说明该程序是一个运行在bash\_shellshock下的shell脚本。myprog.cgi程序被调用的时候，会调用fork函数创建一个新进程，然后再使用exec函数来执行CGI程序，并将环境变量传递给子进程。其中有些环境变量可以通过人为控制传入指定的字符，例如HTTP\_USER\_AGENT。

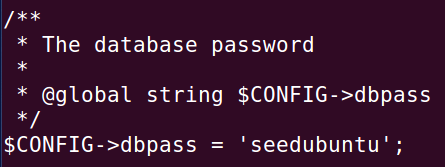
User-Agent / HTTP\_USER\_AGENT并不是唯一可以使用的字段，curl命令还有其他选项可以修改另外一些字段。说明CGI进程会从远程用户的HTTP请求中获取一些信息来赋值给服务器的环境变量。这就导致服务器存在了可以被进攻的漏洞。

**Task 4**

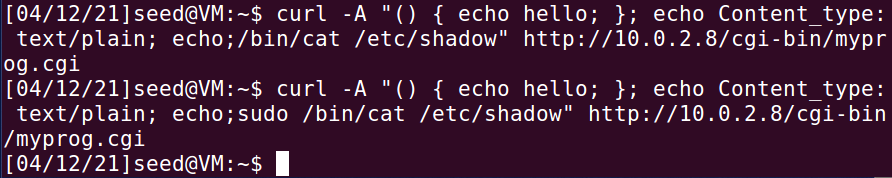
通过shellshock攻击获取一些机密文件的内容：



环境变量USER\_AGENT被传递给子进程，被转化为一个函数和三个shell命令并且执行，从而得到数据库密码：



同样方式尝试查看 /etc/shadow文件：



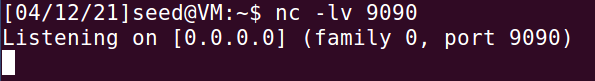
发现无法查看。

推测这是因为/etc/shadow文件的权限为000，需要root权限才可以查看。

**Task 5**

首先来正常构建一个反向shell：

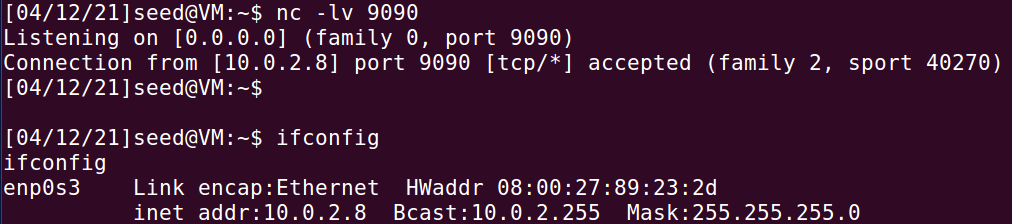
在攻击者主机(10.0.2.7)上，监听9090端口的TCP连接：



在目标主机(10.0.2.8)上，运行如下指令该指令会触发一个连接到攻击者主机9090端口的TCP连接：



攻击者主机收到并建立连接，在攻击者主机(10.0.2.7)上输入ifconfig可以看到，此时的shell运行在目标主机 (10.0.2.8)上，这样就是构建了一个反向shell。

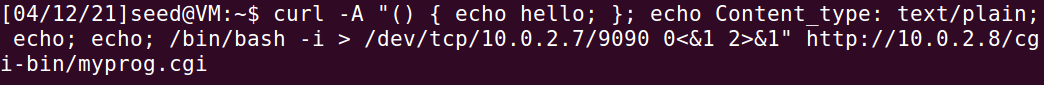


下面使用shellshock攻击来创建一个反向shell：

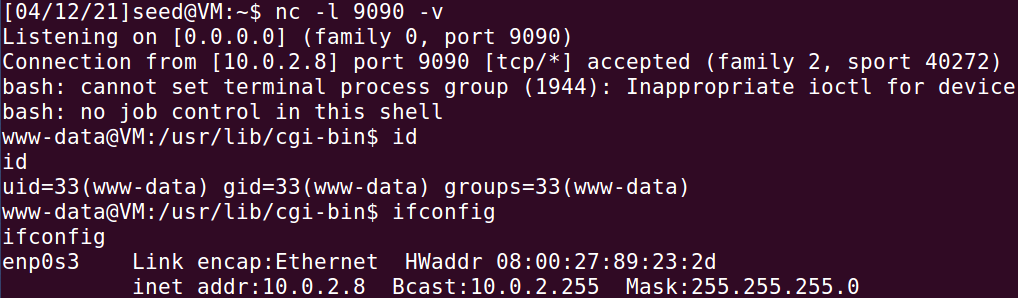
在攻击者主机(10.0.2.7)上，监听9090端口的TCP连接：



在攻击者主机(10.0.2.7)用新终端运行如下指令，进行shellshock攻击：



curl指令被执行，使得目标服务器触发了一个shell并连接到了攻击者主机的9090端口。这个shell的输入来自TCP连接，输出又被传给同一个TCP连接。攻击者的netccat程序接受连接并显示由目标服务器的CGI程序触发的shell程序送来的shell提示符，这样就成功创建了反向shell：

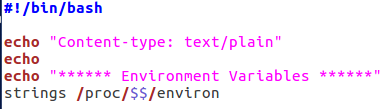


反向shell实际是在目标服务器上运行，但是它会从攻击者主机上获取输入，并shell运行输出打印在攻击者主机上。

**Task 6**

本实验需要使用具有防护补丁的bash程序，只需要将myprog.cgi代码的第一行中的bash\_shellshock改为bash：

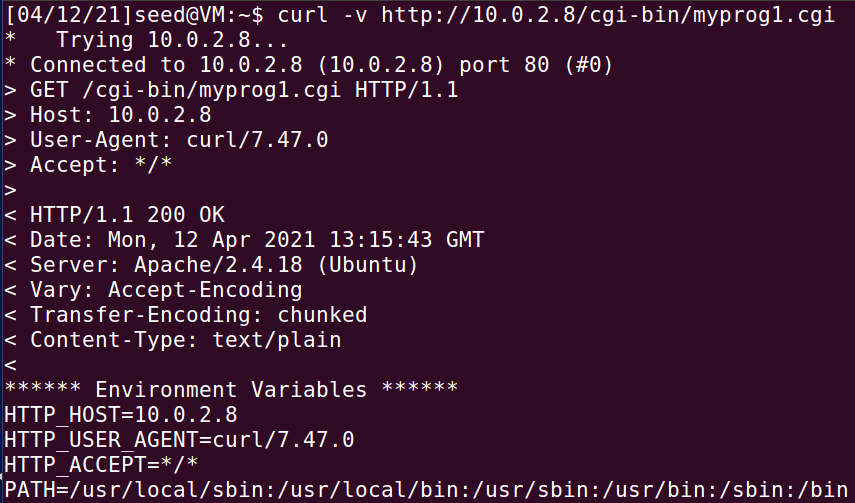
新建文件myprog1.cgi：



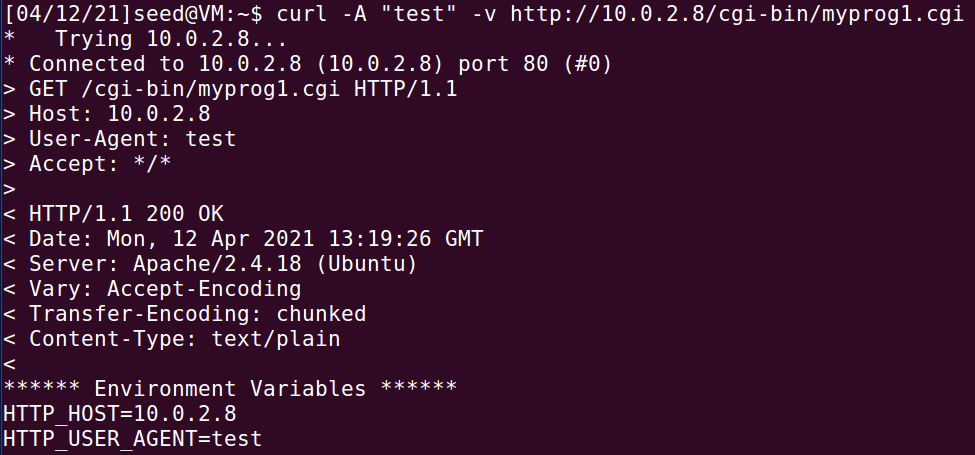
将其文件权限设置为755。

重做task3：

在攻击者主机(10.0.2.7)上，使用curl命令 -v 选项打印出HTTP请求和服务器的响应，注意将myprog.cgi替换成myprog1.cgi：



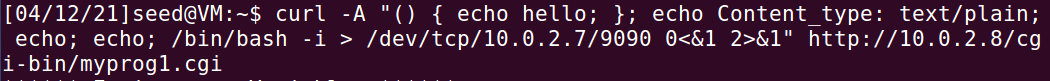
使用curl -A 命令修改HTTP请求的User-Agent字段为“test”：



目标服务器(10.0.2.8)中的环境变量还是会被修改。

重做task5：

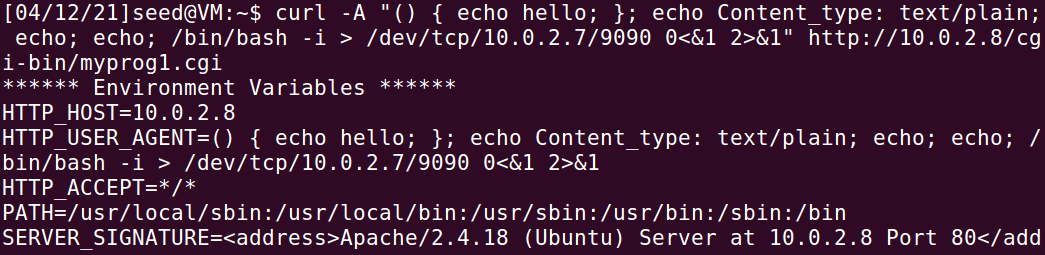
注意将myprog.cgi替换成myprog1.cgi



攻击者主机没有建立连接：



没有成功创建反向shell，而是直接打印出了环境变量：



这应该是因为bash修补了漏洞之后，不会把环境变量解析为函数和命令运行，这样一来CGI程序被执行时虽然还会被以同样的原理提供环境变量，但恶意植入的代码无法发挥作用。