**21CY682 – Secure Coding lab – I**

**Assignment Topic: SHELLSHOCK**

**Register Number: CYS22005**

**Date: 21/11/2022**

**Name: B.Shebu**

**Background : Shell functions –**

Bash shell is one of the most popular shell programs inLinux.

Te shellshock vulnerability are related to previous bas shell versions.

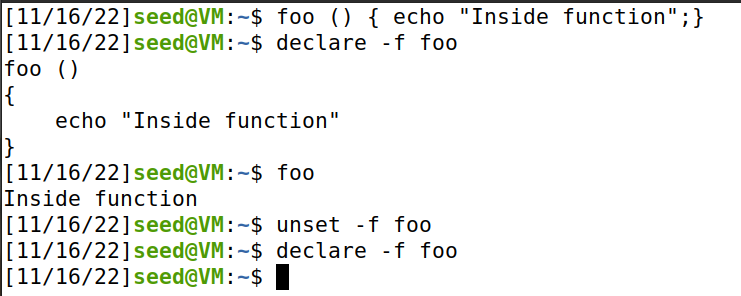
Here in te below screenshot we can see that , we **create a function** called foo with the contents “Inside function”.

When we use the **declare –f** command followed by the function name, it displays the content of the foo() function.

While **running the name of the function foo**, bash displays the output of the function.

While using the **unset command** , we are removing the function declaration tat we just made

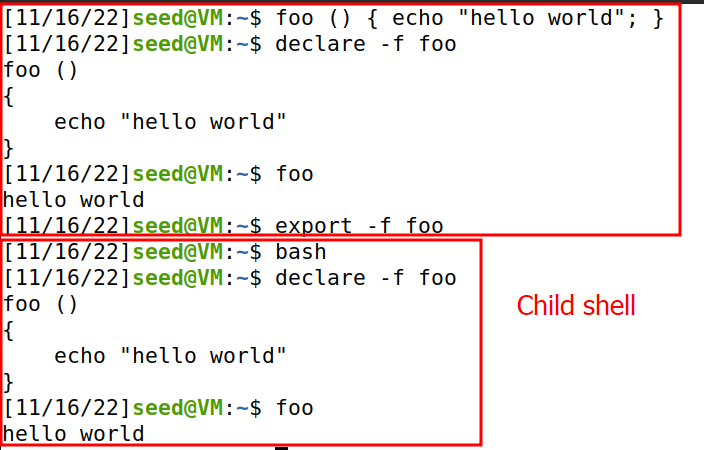
So, now if we again **try to display the contents of the foo() we don’t get any result** since we unset the function declaration.



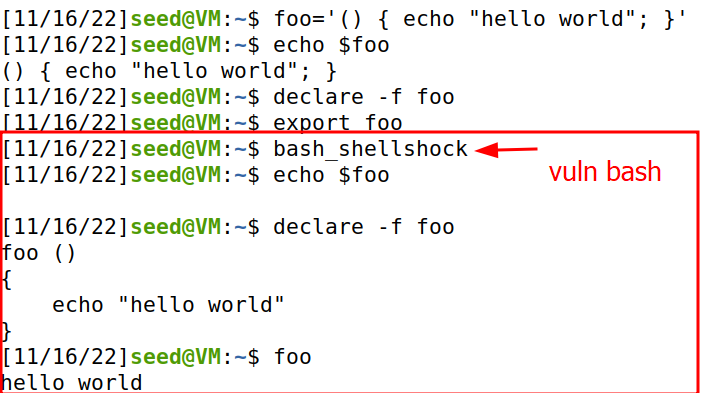
**Passing shell function to child process -**

Here , we repeat the same process in the parent shell but at the end we export the foo() .

Now after running the bash command we create a new child process. Now if we run the declare command we still get the content of that function, which means that the child process will have the contents of the parent process shell since we exported the foo().



**Passing shell function to child process.**



In the parent shell which is not a vulnerable bash version, we declare a variable foo with a value hello world. But after creating a child shell which is the vulnerable bash version for shellshock, when we try to echo the variable value we don’t get the value instead we get the value when we use declare statement.

This is because the vulnerable version of bash considers any thing that matches with “() {” as a function.

**SHELLSHOCK Vulnerability –**

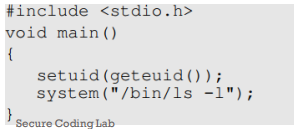
Vulnerability named Shellshock or bashdoor was publicly release on September 24, 2014. This vulnerability was assigned CVE-2014-6271

This vulnerability exploited a mistake made by bash when it converts environment variables to function definition .

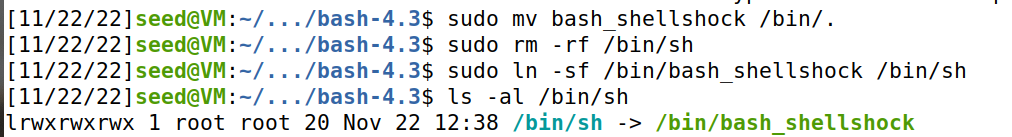
Here parent process can pass a function definition to child process via environmental variable. Due to a bug in parsin logic, bash executes some part of command contained in the variable.

In the below screenshot we can see that when we invoke a child process which is vulnerable version of shellshock, it execute some part of extra command at the end. 

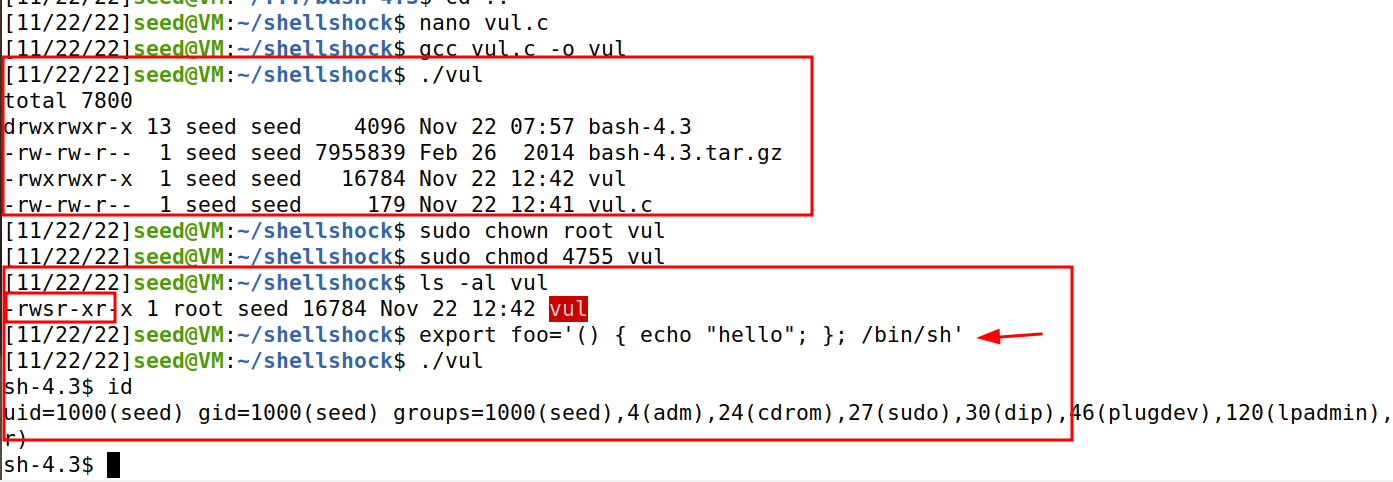
**Shellshock attack on SET-UID programs**



The above program uses system() function to run /bin/ls command.



Here in the above screenshot , we have removes sh binary from /bin directory and made a soft link of bash\_shellshock pointing to /bin/sh.

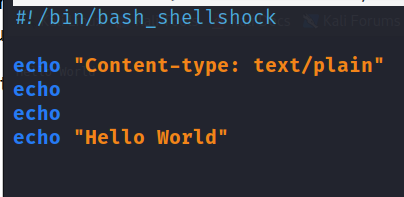


While running the program, we see that when the the program is set as SETUID , it will invoke /bin/sh as root . So as per shellshock vulnerability , it execute the extra part . If it is a sell command the function will execute it.

So we got a privileged shell on running the program vul which is a setuid program.

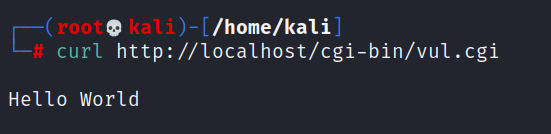
**SHELLSHOCK ATTACK ON CGI PROGRAMS –**

Here we write a very simple cgi program (vul.cgi) which just prints out the contents “Hello World” & store it in the location /usr/lib/cgi-bin/.

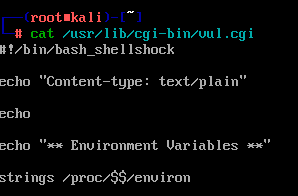


After saving the file , we need to start the apache2 server by the command systemctl start apache2.

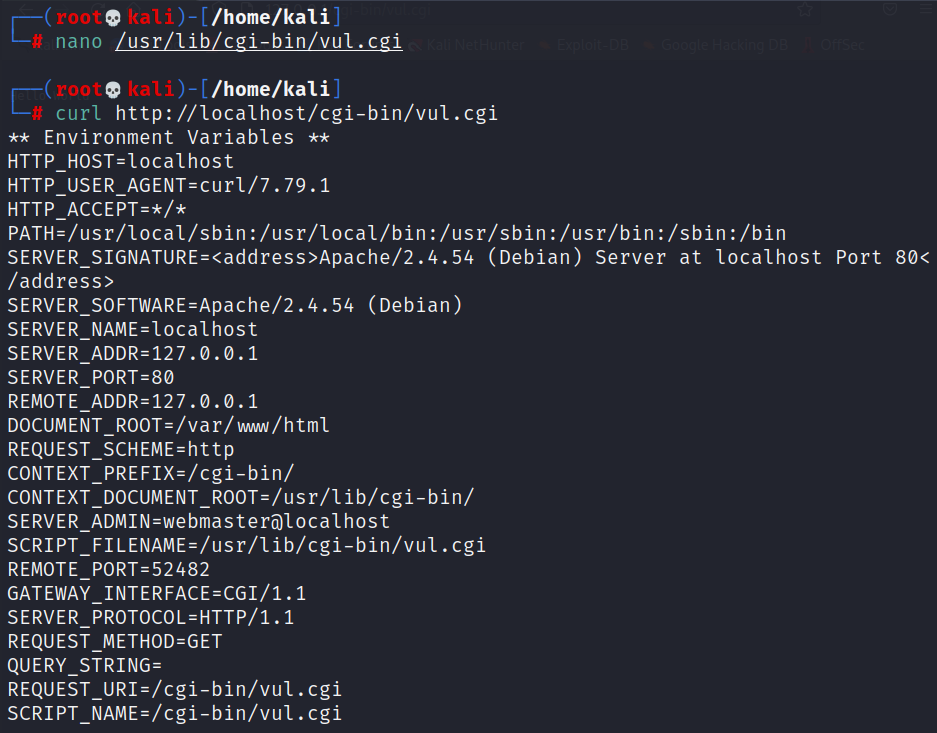
Now when we use curl command to the url in localhost to run the cgi script. We get the output Hello World as below.



Here we modify the vul.cgi to this program which will print out all the env variables of the server running vulnerable version of shellshock .

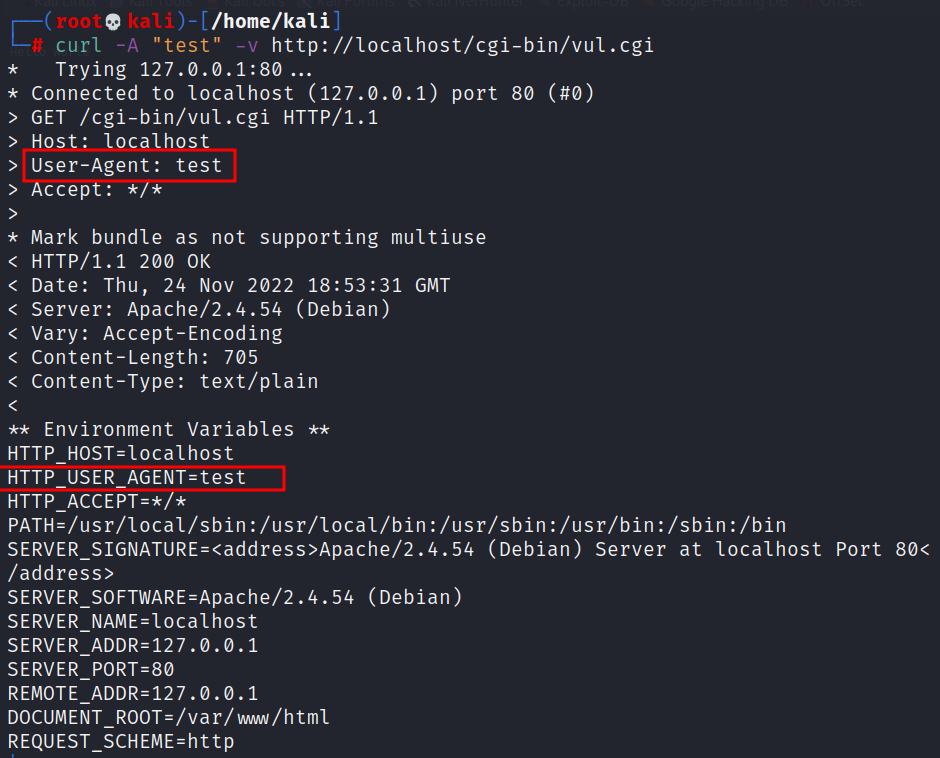


Now if we use curl command on the url ,it executes the cgi script and sends us the environmental variables. Here the **user agent is curl/7/79/1 .**



In curl command , we can use the **–A flag to change the user-agent field** to whatever we want. So we give **–A “test”.**

As seen below in the request and response, we see that the USER\_AGENT is set to test.



**Launching the shellshock attack**

Now with the below payload , we were able to execute /bin/ls command on the server using shellshock vulnerability thus leading to remote code execution on the server which is pretty dangerous.

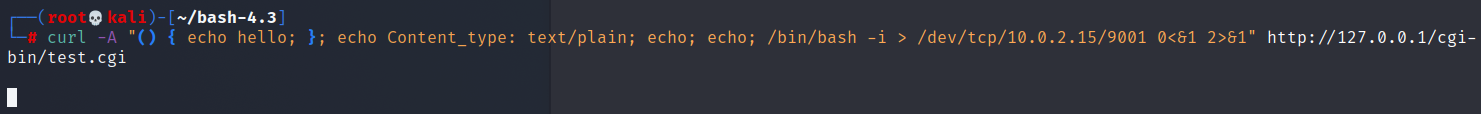


**Gaining a reverse shell (RCE) from the server** –

▪ Attackers like to run the shell program by exploiting the shellshock vulnerability, as this gives them access to run whichever commands they like.

▪ Instead of running /bin/ls, we can run /bin/bash. However, the /bin/bash command is interactive.

▪ If we simply put /bin/bash in our exploit, the bash will be executed at the server side,



▪ We start a netcat (nc) listener on the Attacker machine .

▪ We run the exploit on the server machine which contains the reverse shell command ▪ Once the command is executed, we see a connection from the server .

