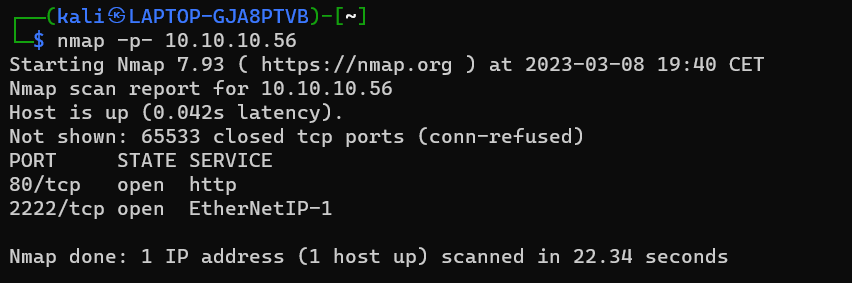
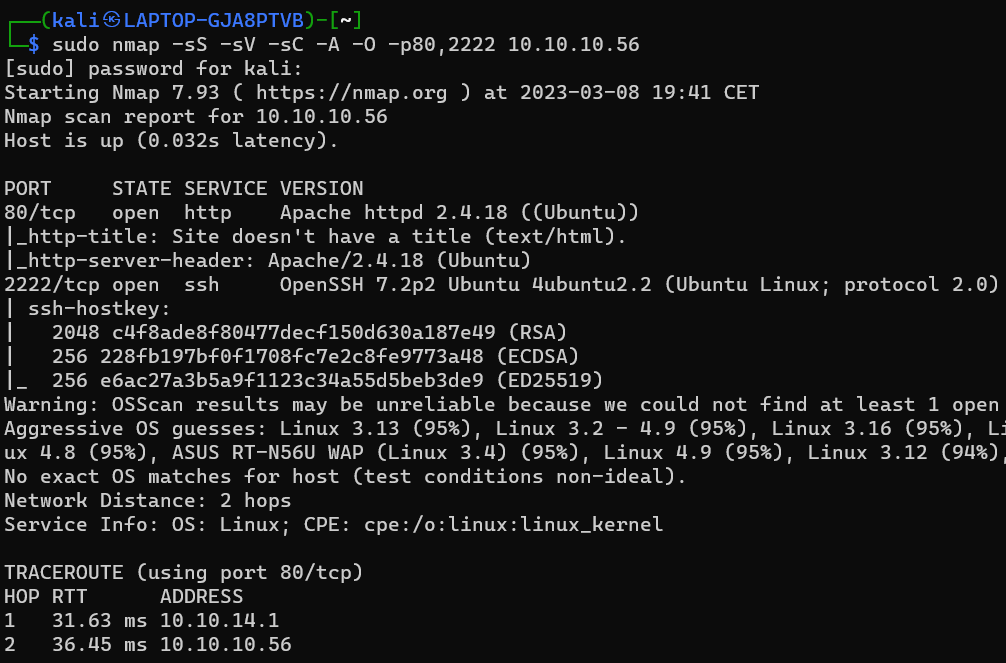
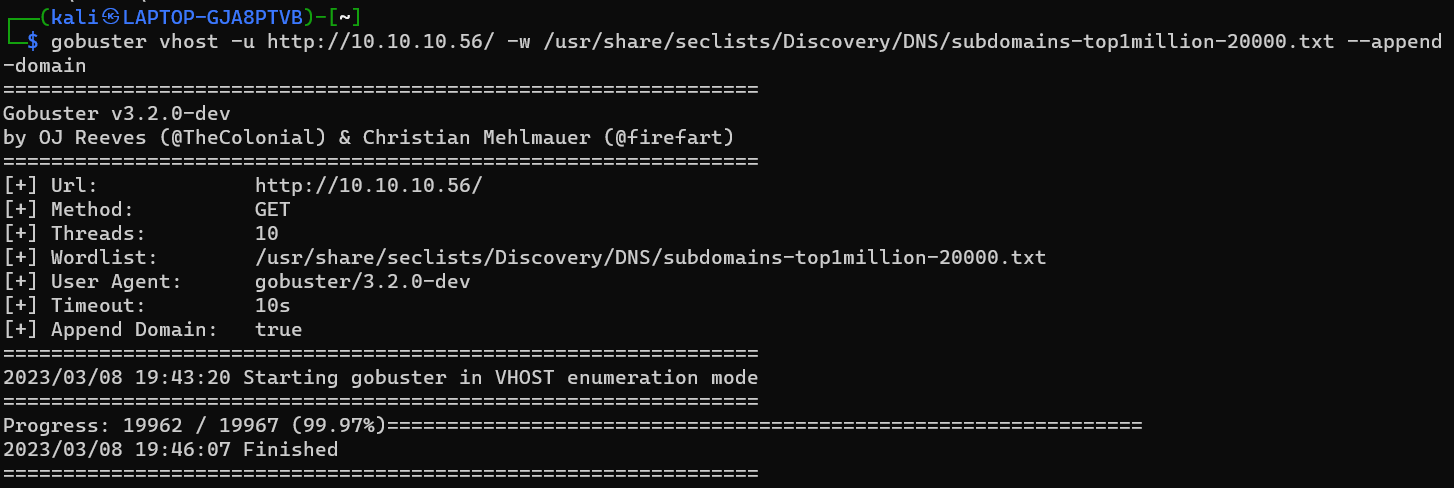
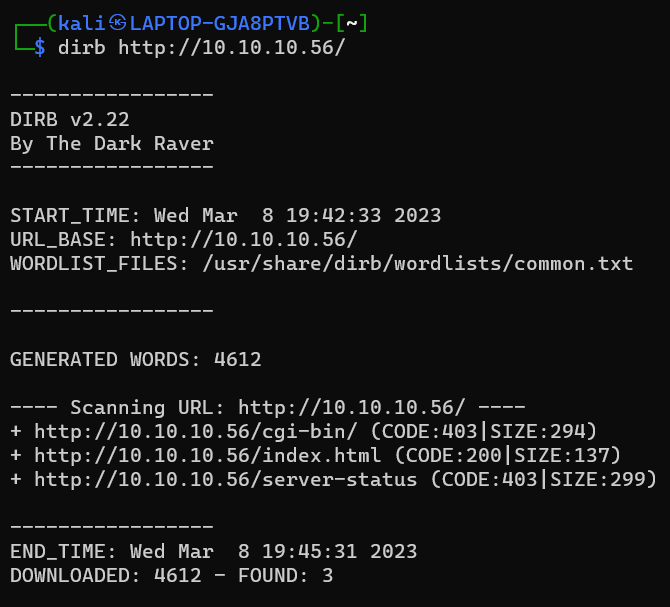
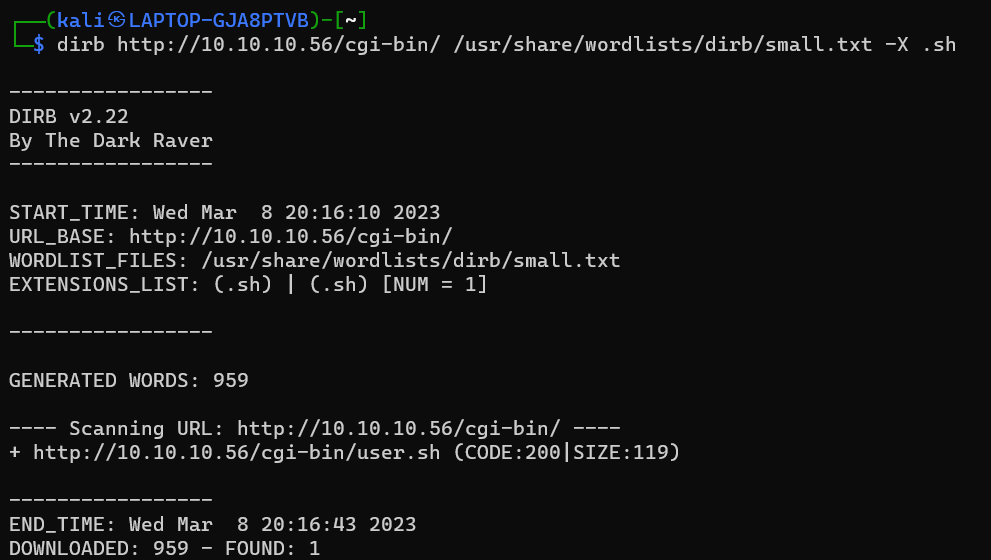
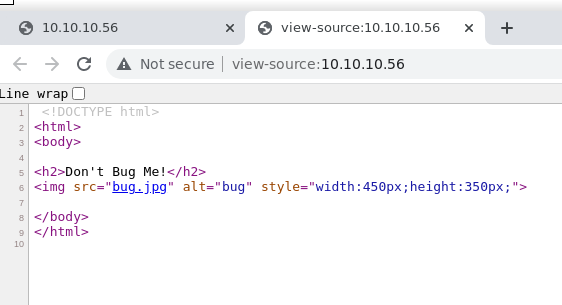
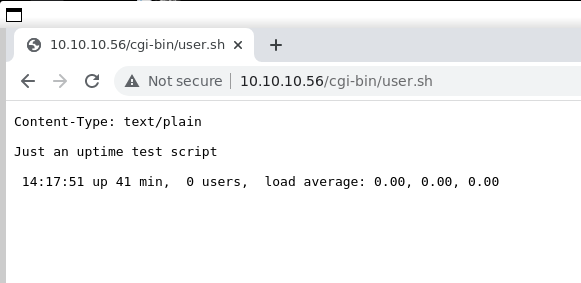
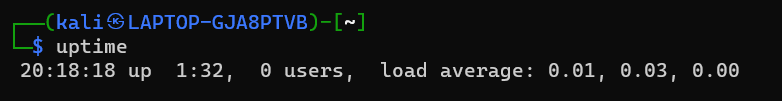
1. Pinged
2. Nmap -p-
3. Nmap -sS -sV -sC -A -O -p80,2222
4. Let’s check for subdomains with gobuster
5. Let’s enumerate directories with dirb
6. We’ve found a dir named cgi-bin. Always run dirb on those dirs cause they might contain files.
7. We’ll run dirb again, but this time including the /cgi-bin. We’re also gonna specify some extensions like sh,cgi and pl



1. We’ve found a file! Time to go and check the website with chromium



1. As we can see, the website is extremely simple. Lets go check the file we’ve found tho
2. So it look like a script that is being executed by bash. The specific command is uptime:



1. So when we see something like this: a website executing a script on the system. We need to test the target against the http shellshock vuln, but first what is this vuln now?.

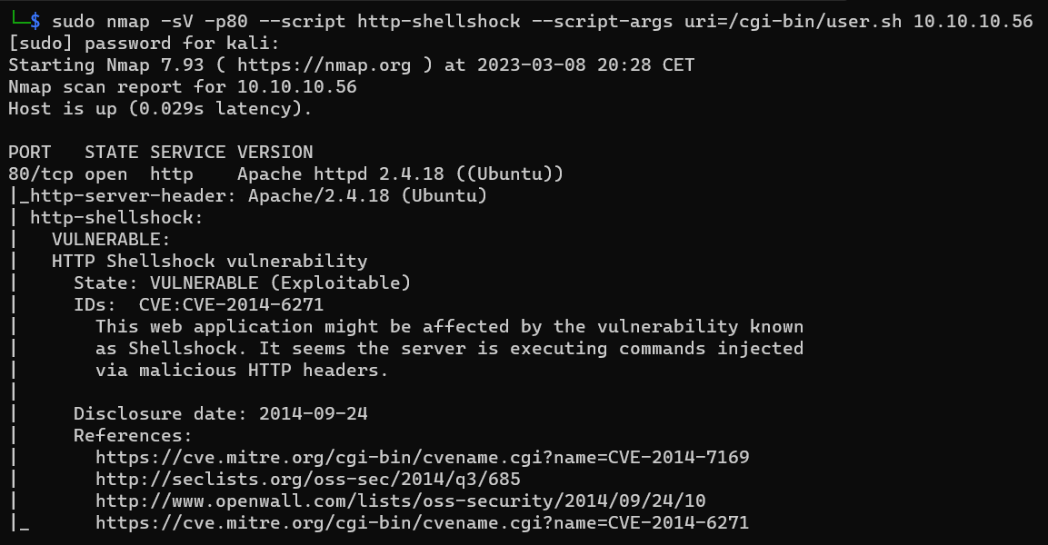
The shellshock vulnerability, is a web vulnerabilities that affects web servers that use (cgi/sh/pl/other) scripts to handle http requests and use an outdated bash version (1.14 – 4.3). In these bash versions is a flaw in the way that bash handles env variables. To be specific, bash allows environment variables to be set with arbitrary code that is executed when the variable is referenced in a bash command.

Here is how http shellshock works:

1. We send an http request to the vulnerable web server, including some malicious code inside the user-agent header.
2. The web server receives the request, it processes the user-agent header, and stores it in an environment variable

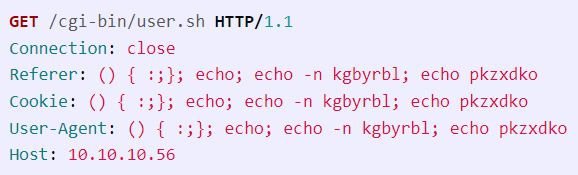
Now this happens thus when the website runs the (cgi/sh/pl) script that handles the http request.

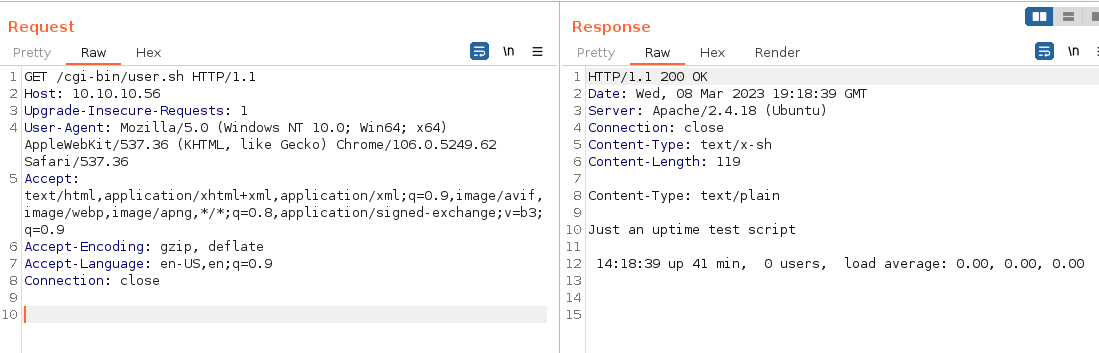
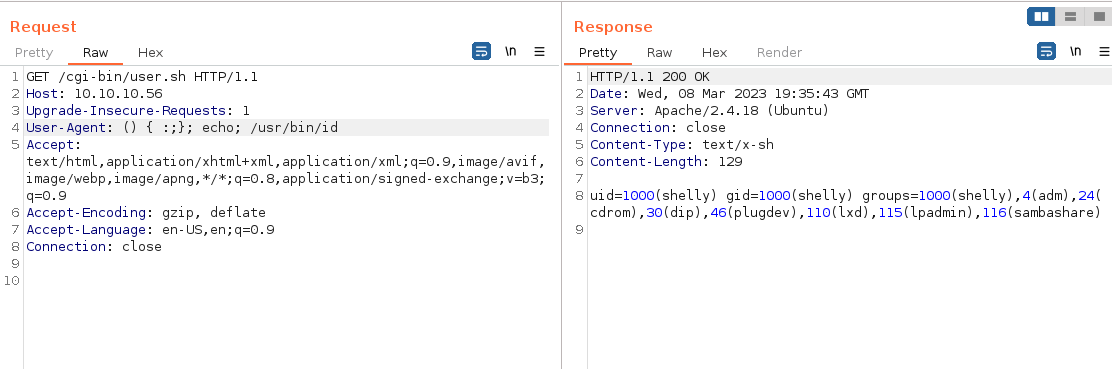
1. But when the script runs, and it references the user-agent variable, the code that we set there, will get executed on the system.
2. So, in summary: when a CGI script written in Bash is executed, Bash reads some environment variables that contain information about the HTTP request. If the user-agent header contains a Bash function definition with malicious code in it, Bash will run that code when it defines the function. This can be used by attackers to run code on the server that the CGI script is running on, which can be a serious security problem.
3. Lets see if the webserver is vulnerable to it:



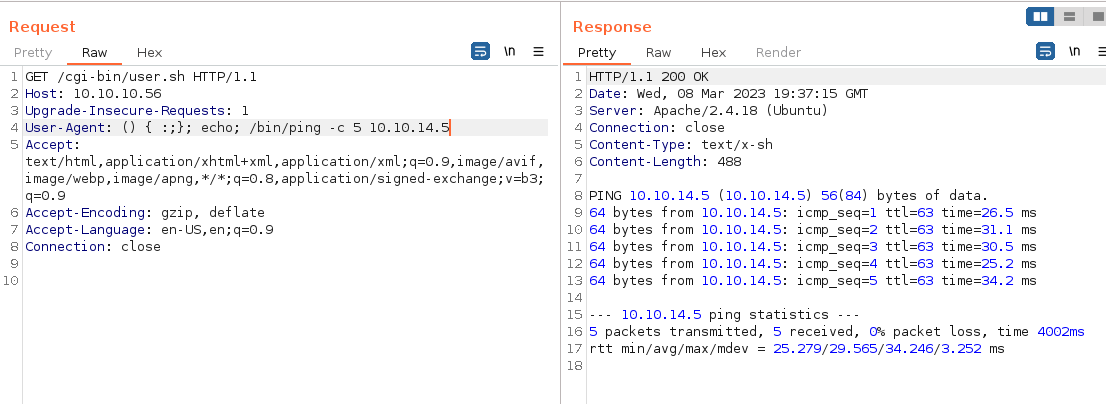
And of course, it is in fact vulnerable!

This is how nmap performed the test:



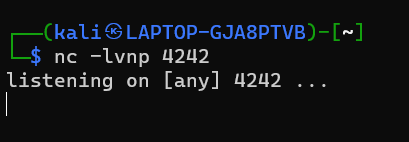
1. Lets open burpsuite, capture the request, and send it to the repeater:
2. Now lets put some payloads in the user-agent and see what we get:

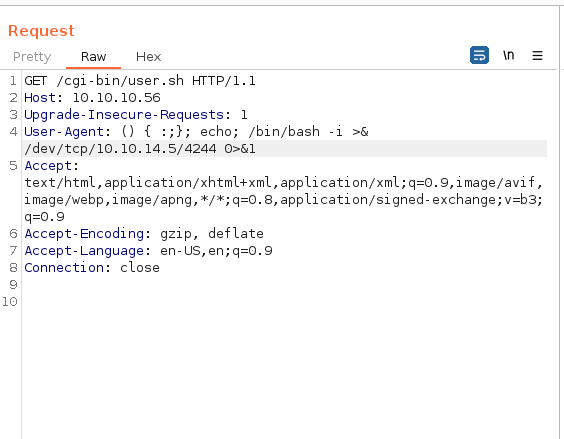
It works!

1. Lets see if we can ping our own system, because if that’s possible, a reverse shell is possible:

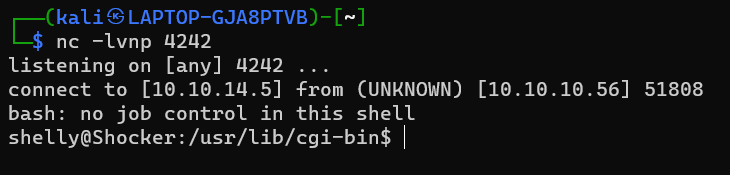
And that also works!

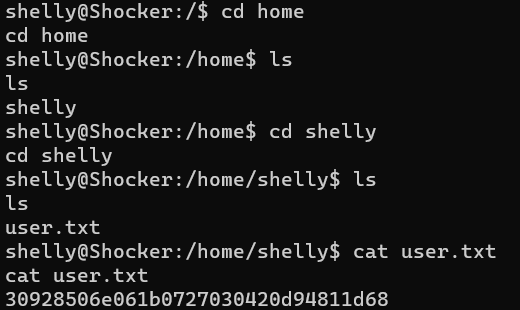
1. Lets craft a nice bash reverse shell, first start a listener





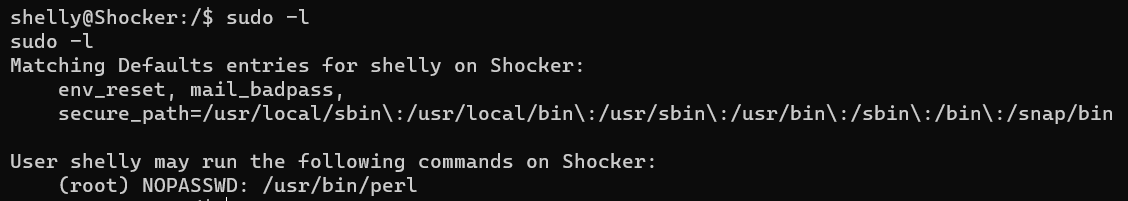
1. .

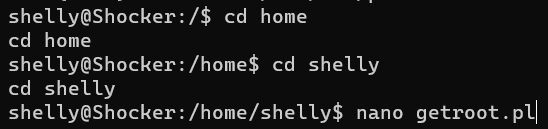
Aaaand we have a shell:

1. Grab user.txt

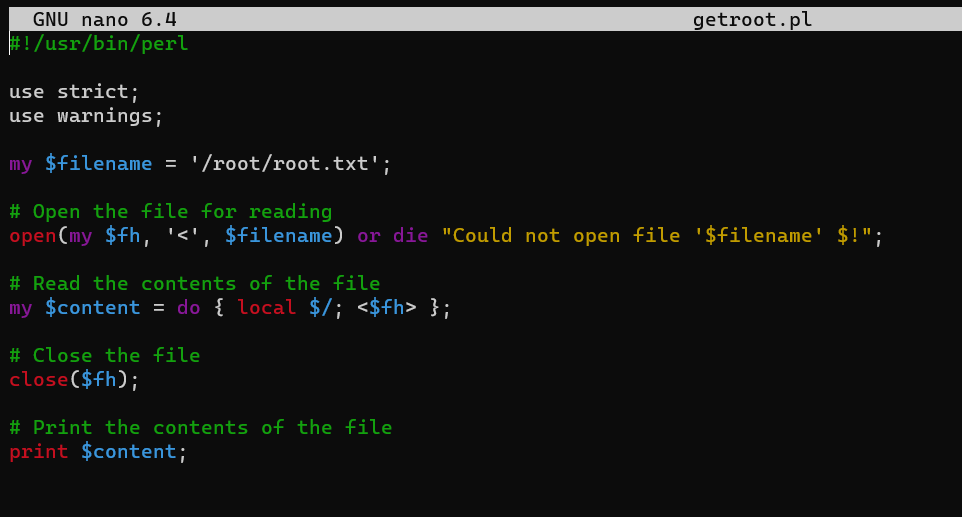
Now we only need root privileges to enter the root directory and grab root.txt. but before we upload any enumeration scripts and try to escalate privileges. Lets just try sudo -l, to see if there are any command we can use with sudo without a password.

1. Sudo -l

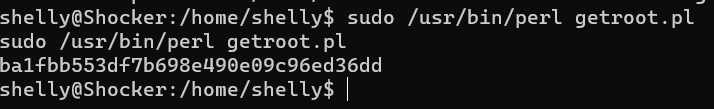


Yess, now we only need to create a perl scripts that opens /root/root.txt, and prints out the value of it. Easy enough:

1. .



1. And now we run it with sudo:



And here we are. I’ve done it again.