TREND MICRO™

Securing Your Journey to the Cloud

TECHNICALBRIEF

Shellshock: A Technical Report

Trend Micro Threat Research Lab

Introduction

On September 24, 2014, Stephane Chazelas discovered that Bash incorrectly handled trailing code in function definitions, as described in CVE-2014-6271.[1] Attackers can exploit what has been dubbed "Shellshock" to bypass environment restrictions. Several programs such as Secure Shell (SSH) and Common Gateway Interface (CGI) scripts allow Bash to run in the background. Because of this, attackers can remotely exploit the vulnerability, making Shellshock a serious threat. Researchers even warn that it can become as big as Heartbleed, also known as CVE-2014-0160, which was discovered this April.[2]

After performing tests, we found that not every system can be remotely exploited because they run Bash. Systems also need to run applications that make Bash accessible over the network to become vulnerable.

This report provides an in-depth technical description of Shellshock.

Vulnerability Details

In the next few days, more information about Shellshock will be disclosed and it can be completely patched. As previously mentioned, CVE-2014-6271 or Shellshock was originally discovered by Stephane Chazelas, a Unix and Linux network and telecommunications administrator and IT manager at a U.K.-based robotics company, SeeByte Ltd.[3]

Proof of Concept

Running the following command is a simple test to check if Bash is vulnerable on a system:

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

Running the command above on a system running a vulnerable Bash version will respond with the output, "vulnerable." The patch issued to fix the problem ensures that code is not allowed to run after any Bash function.

How the Vulnerability Can Be Exploited

We used the proof-of-concept (PoC) code to see how the vulnerability could be exploited.

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

The code above declares an environment variable then prints "test." Due to the specially crafted value of the environment variable, Bash takes the code as a function definition as in:

```
x() { :;
};
echo vulnerable
```

Once a system is declared "vulnerable," attackers can then execute a malicious command after the function definition.

Impact

Bash is a widely used Unix shell in several systems and software and Shellshock affects versions 1.14–4.3 of GNU Bash, specifically:

- Older than Bash 4.3 patch 25
- Older than Bash 4.2 patch 48
- Older than Bash 4.1 patch 12
- Older than Bash 4.0 patch 39
- Older than Bash 3.2 patch 52
- Older than Bash 3.1 patch 18
- Older than Bash 3.0 patch 17
- Bash 2.0.5 and older
- Bash 1.14.7 and older

Applications and networked devices that use Bash, including routers, IP cameras, gateways (e.g., Citrix's NetScaler, F5's BIGIP, and Cisco products), and Web CGI programs are vulnerable. Attackers can even command vulnerable Dynamic Host Configuration Protocol (DHCP) server to execute arbitrary code on client systems.

The following table lists commonly used OSs and their default shells. OSs whose default shell is Bash are vulnerable.

OS	DEFAULT SHELL
Mac OS X	Bash
RHEL	Bash
CentOS	Bash
Fedora	Bash
Debian	sh (lenny) dash (Squeeze)
Ubuntu	dash
FreeBSD	tcsh
Android	Newer releases: ash; mksh is shipped with Android 3.0
iOS (only on jail-broken devices)	Bash
Embedded devices	Mostly use BusyBox (ash)

Attack Scenarios

Web Server Attack

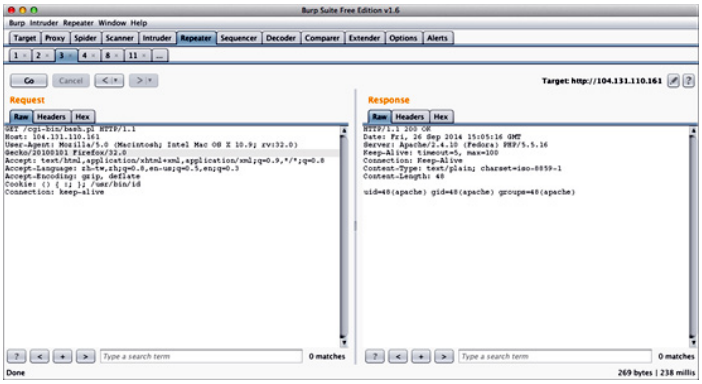
Web server attacks are most problematic as Bash scripts are executed via cgi-bin.

In this scenario, CGI requires a Web server to convert HTTP request headers and pass them on from a client to environment variables. This is performed by the *mod-cgi* or *mod_cgid* module of an Apache server. If attackers call a Bash script via cgi-bin, they can use it to execute code as an httpd with the Web server's permission.

We tested this out on the Apache *mod_cgi* module using different programming languages. Findings showed that CGI programs written in Perl, PHP, Python, and Ruby are vulnerable.

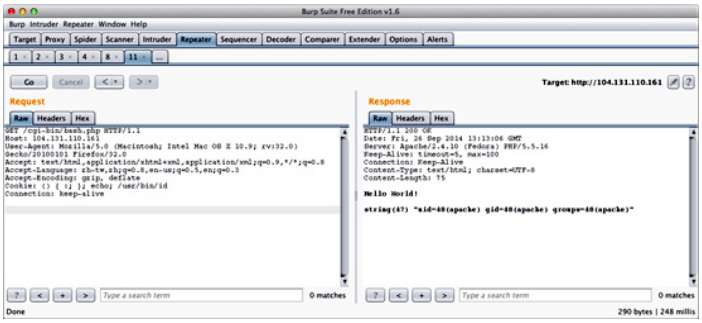
Perl

```
#!/usr/bin/env perl
print "Contenttype: text/plain; charset=iso88591\n\n";
$result = system("test >> /dev/null;");
```



PHP

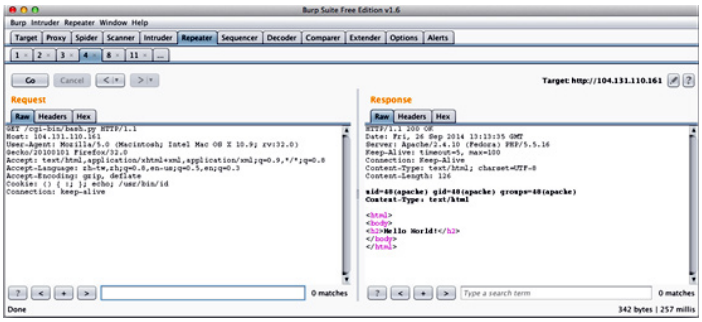
```
#!/usr/bin/env php
<?php
print "ContentType: text/html\n\n";
print "Hello World!\n\n";
var_dump(exec('date'));
?>
```



Python

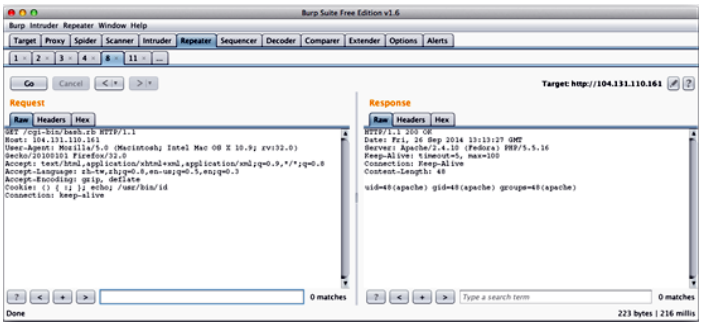
```
#!/usr/bin/env python
print "ContentType: text/html"
print
print """
<html>
<body>
<h2>Hello World!</h2>
</body>
</html>
"""

import os
os.system('date')
```



Ruby

```
#!/usr/bin/env ruby
puts "Contenttype: text/html\n\n"
puts "<?xml version=\"1.0\" encoding=\"UTF8\"?>"
puts "<!DOCTYPE html>"
puts "<html>"
puts "<head>"
puts "<title>Ruby CGI test</title>"
puts "</head>"
puts "<body>"
puts "<p>Hello, world!</p>"
puts "</body>"
puts "</html>"
`date`
```

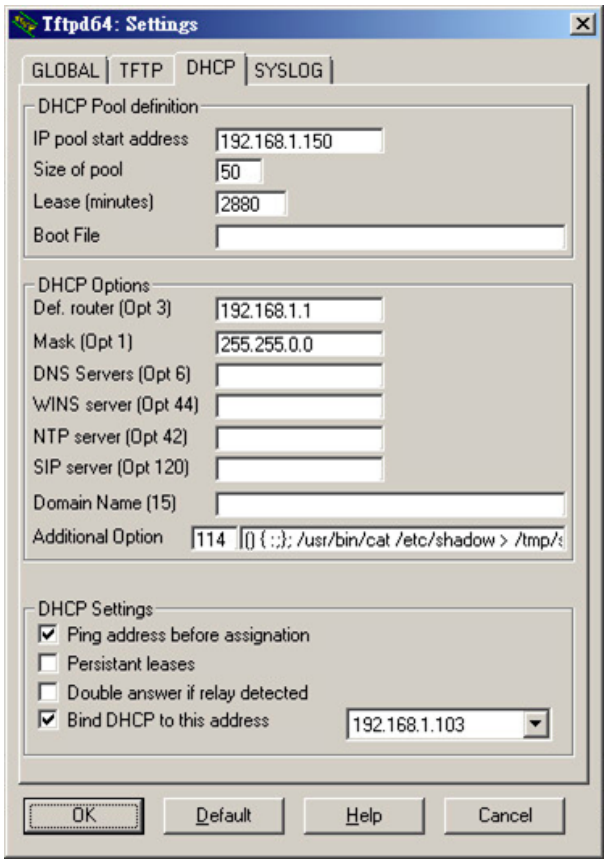


DHCP Server Abuse

Exploiting a vulnerable DHCP server can allow attackers to spread arbitrary commands to clients connected to local network environments. They can set up malicious DHCP servers with specially crafted command options to execute various malicious payloads such as:

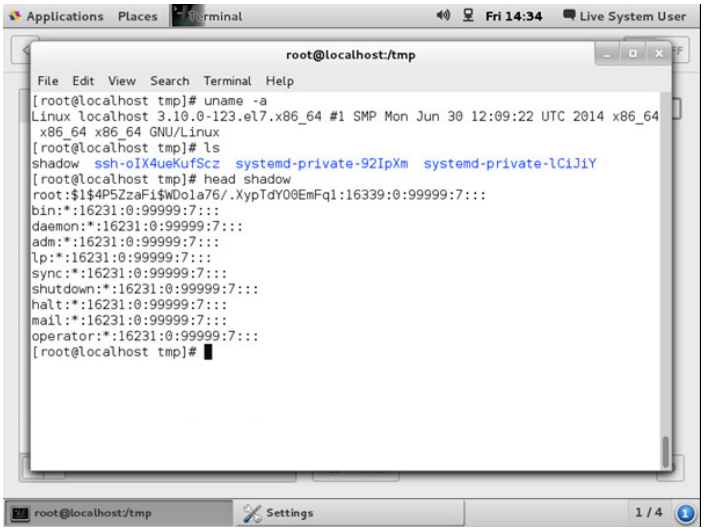
```
() { :; }; /usr/bin/cat /etc/shadow > /tmp/shadow
```

The following is a sample malicious DHCP server setup, which has been configured with a malicious payload in Additional Option 114:

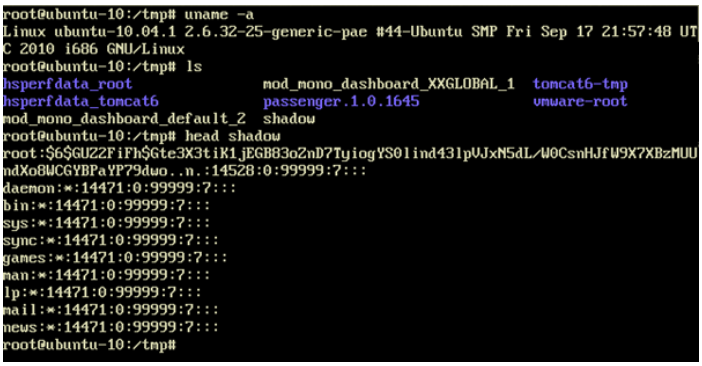


A user who accesses the malicious server's IP address will execute the malicious command on his system. The command will then move the shadow file to */tmp/* as shown in the following:

- On a CentOS 7.0 DHCP client running a vulnerable Bash version

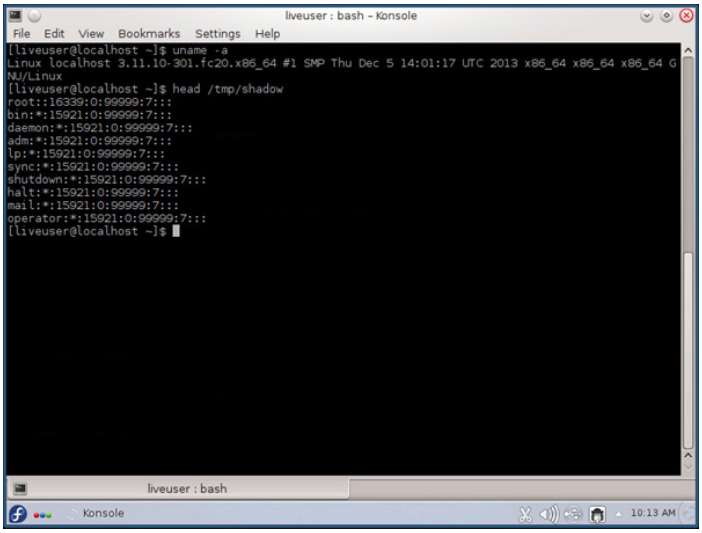


- On an Ubuntu 10.04.1 LTS DHCP client running a vulnerable Bash version



- On an Ubuntu 14.04.1 LTS DHCP client running a vulnerable Bash version

- On a Fedora 20 DHCP client running a vulnerable Bash version



Mobile devices that run Android and iOS are not vulnerable because they do not come shipped with Bash by default. Mac OS X is not vulnerable as well because it does not use Bash when requesting for IP addresses during the DHCP process.

OS	VERSION	STATUS
CentOS	7.0	Vulnerable
Fedora	20	Vulnerable
Ubuntu	10.04.1 LTS	Vulnerable
Ubuntu	14.04.1 LTS	Vulnerable
Android	4.4.4	Invulnerable
iOS	7.0.4	Invulnerable
Mac OS X	10.9.5	Invulnerable

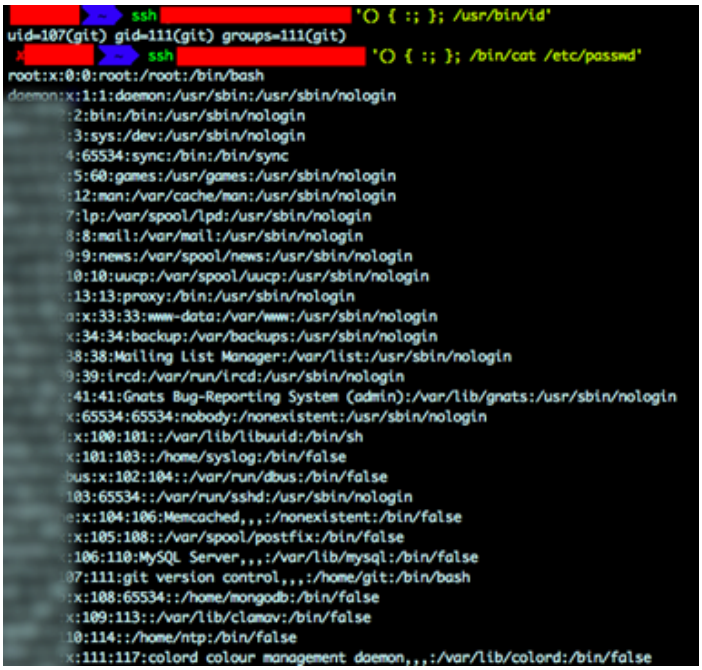
GIT or Subversion Server Attack

Attacking GIT or subversion servers can give attackers access to connected systems or servers but not the ability to execute arbitrary commands due to a restricted shell environment. CVE-2014-6271, however, allows them to bypass restrictions and get shells to work. (Regular OpenSSH users are not affected because they already have shell access.)

Vulnerable servers whose default GIT user shell is Bash are vulnerable to remote code execution, especially those with uploaded SSH keys that attackers can get their hands on. Note that only GIT servers that use Bash for default shells are vulnerable. Those that do not are not vulnerable such as dash on Debian.

We used the following PoC Shellshock code to obtain a vulnerable restricted GIT server's password:

ssh git@gitserver '() { :;}; echo vulnerable'



Other Attack Scenarios

Even though this report presents a limited number of exploit scenarios, several potential possibilities can still emerge. But we know one thing for sure, the vulnerability can spread via worms. In fact, we have already seen worms spread Shellshock.

Solutions and Recommendations

Companies should test their Linux-based servers, devices, and third-party applications that allow Bash to run in the background. They should especially test Web-based applications and services that attackers can easily remotely exploit with Shellshock. Linux versions such as Xymon, a very popular server or network monitoring system are vulnerable to Shellshock. We strongly recommend Bash users to update to the latest versions as shown in the following table:

OS	AVAILABLE INVULNERABLE VERSION
RHEL	RHSA-2014:1306-1 BASH-3.2-33.el5_11.4 (RHEL5) BASH-4.1.2-15.el6_5.2 (RHEL6) BASH-4.2.45-5.el7_0.4 (RHEL7)
Fedora	BASH-4.2.48-2.fc19 (Fedora 19) BASH-4.2.48-2.fc20 (Fedora 20) BASH-4.3.25-2.fc21 (Fedora 21)
CentOS	BASH-3.2-33.el5_10.4 (CentOS 5) BASH-4.1.2-15.el6_5.2 (CentOS 6) BASH-4.2.45-5.el7_0.4 (CentOS 7)
AWS	CVE-2014-6271 Advisory ALAS-2014-418
Ubuntu	USN-2362-1 (CVE-2014-6271) USN-2363-1 (CVE-2014-7169) 4.1-2ubuntu3.2 (Ubuntu 10.04 LTS) 4.2-2ubuntu2.3 (Ubuntu 12.04 LTS) 4.3-7ubuntu1.2 (Ubuntu 14.04 LTS)
SuSE	CVE-2014-6271/Bug 896776 CVE-2014-7169/Bug 898346 3.2-147.20.1 (SuSE11) 3.1-24.32.1 (SuSE10)
Debian	DSA-3032-1 (CVE-2014-6271) DSA-3035-1 (CVE-2014-7169) 4.1-3+deb6u2 (squeeze (lts)) 4.2+dfsg-0.1+deb7u3 (wheezy(security)) 4.3-9.2 (sid)
Gentoo	Bug 523592 BASH-3.1_p18-r1 BASH-3.2_p52-r1 BASH-4.0_p39-r1 BASH-4.1_p12-r1 BASH-4.2_p48-r1

OS	AVAILABLE INVULNERABLE VERSION
Scientific Linux	SLSA-2014:1293-1 BASH-3.2-33.el5.1(SL5) BASH-4.1.2-15.el6_5.1(SL6)

A patch for the Bash vulnerability has been released but it remains incomplete. Some believe the fix still allows certain characters to be injected in vulnerable Bash versions via specially crafted environment variables. Attackers can still craft new methods to bypass environment restrictions and execute shell commands. Bypass methods identified in the following still work:

- CVE-2014-7169[4]
- CVE-2014-7186[5]
- CVE-2014-7187[6]
- https://rhn.redhat.com/errata/RHSA20141306.html

Companies should keep an eye out for updates even if they have already patched CVE-2014-6271. They should also implement the following additional measures to protect their networks:

- Switch default shell to a Bash alternative such as dash or tsh. (Do so carefully, however, because different shells use different syntaxes. This can break existing scripts.)
- To protect CGI programs, filter requests using the following intrusion detection system (IDS) or Web application firewall rules to block exploits:
    - mod_security
        - Request header values:

SecRule REQUEST_HEADERS "^\(\) {" "phase:1,deny,id:1000000,t:urlDecode,status:400,log,msg:'CVE-2014-6271 - BASH Attack'"

        - SERVER_PROTOCOL values:

SecRule REQUEST_LINE "\(\) {" "phase:1,deny,id:1000001,status:400,log,msg:'CVE-2014-6271 - BASH Attack'"

        - GET/POST names:

SecRule ARGS_NAMES "^\(\) {" "phase:2,deny,id:1000002,t:urlDecode,t:urlDecodeUni,status:400,log,msg:'CVE-20 14-6271 - BASH Attack'"

        - GET/POST values:

SecRule ARGS "^\(\) {" "phase:2,deny,id:1000003,t:urlDecode,t:urlDecodeUni,status:400,log,msg:'CVE-2014-6271 - BASH Attack'"

        - Filenames for uploads:

SecRule FILES_NAMES "^\(\) {" "phase:2,deny,id:1000004,t:urlDecode,t:urlDecodeUni,status:400,log,msg:'CVE-2014-6271 - BASH Attack'"

    - IPTables

iptables A INPUT m string algo bm hexstring '|28 29 20 7B|' j DROP

ip6tables A INPUT m string algo bm hexstring '|28 29 20 7B|' j DROP

    - Suricata

alert http $EXTERNAL_NET any > $HOME_NET any (msg:"Possible CVE20146271 BASH Vulnerability Requested (header)"; flow:established,to_server; content:"() {"; http_header; threshold:type limit, track by_src, count 1, seconds 120; sid:2014092401;)

    - Snort

alert tcp $EXTERNAL_NET any > $HOME_NET $HTTP_PORTS (msg:"Possible CVE20146271 BASH Vulnerability Requested (header) "; flow:established,to_server; content:"() {"; http_header; threshold:type limit, track by_src, count 1, seconds 120; sid:2014092401;)

- ref:
    - mod_security, IPTables (https://access.redhat.com/articles/1212303)
    - Snort, Suricata (http://www.volexity.com/blog/?p=19)

- Restrict access by source IP address. If CGI program use is still required and patching is not an option, restrict access by source IP address. Only allow trusted IP ranges or hosts to access services.

References

1. NIST. (September 24, 2014). *National Vulnerability Database.* "Vulnerability Summary for CVE-2014-6271." Last accessed September 27, 2014, http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-6271.

2. Trend Micro Incorporated. (2014). *TrendLabs Security Intelligence Blog.* "Heartbleed." Last accessed September 27, 2014, http://blog.trendmicro.com/trendlabs-security-intelligence/?s=heartbleed&Submit=+Go+.

3. SeeByte Ltd. (2014). *SeeByte.* "About SeeByte." Last accessed September 27, 2014, http://www.seebyte.com/about-seebyte/.

4. NIST. (September 24, 2014). *National Vulnerability Database.* "Vulnerability Summary for CVE-2014-7169." Last accessed September 27, 2014, http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-7169.

5. Debian Security. (2014). *Debian.* "CVE-2014-7186." Last accessed September 27, 2014, https://security-tracker.debian.org/tracker/CVE-2014-7186.

6. Debian Security. (2014). *Debian.* "CVE-20147187." Last accessed September 27, 2014, https://security-tracker.debian.org/tracker/CVE-2014-7187.

Securing Your Journey to the Cloud

10101 N. De Anza Blvd.
Cupertino, CA 95014

U.S. toll free: 1 +800.228.5651
Phone: 1 +408.257.1500
Fax: 1 +408.257.2003