Midterm Report of Linux Programming A Linux big Bug – Bash Shellshock

Zhibin Xie, 6103115112, CS153

April 29, 2018

Abstract—A Linux big Bug - Shellshock which was disclosed on 24 September 2014.

Index Terms—Linux, Bash, Bug, Loophole

1 Introduction

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.

–Wikipedia

2 About This Bug

Name: Bash Shellshock
Threat Level: A
CVE: CVE-2014-6271
Discoverer: Stphane Chazelas (France)
Announced: September 25, 2014
Affected System: bash 1.14 to bash 4.3 Linux/Unix System

3 Background

Stphane Chazelas contacted Bash's maintainer, Chet Ramey, on 12 September 2014[1] telling Ramey about his discovery of the original bug, which he called "Bashdoor". Working together with security experts, he soon had a patch as well. The bug was assigned the CVE identifier CVE-2014-6271. It was announced to the public on 24 September 2014 when Bash updates with the fix were ready for distribution.

Shellshock errors affect Bash, a variety of Unix-based systems used to execute command-line and command scripts. It is usually installed as the system's default command line interface. An analysis of Bash's source code history shows that these bugs have existed since the Bash version 1.03 was released in September 1989.

Shellshock is a privilege escalation vulnerability that provides system users with ways to execute commands that should not be available. This happens through Bash's "function export" function, so the command script created in a running Bash instance can be shared with subordinate instances. This is achieved through an in-table encoding script (called an environment variable list) that is shared between instances.

Each new instance of Bash scans this table to get the encoded script, assembles each instance into a command that defines the script in a new instance, and then executes the command. The new instance assumes that the script found in the list comes from another instance, but it cannot verify this, nor can it verify that the command it builds is a properly formed script definition. Therefore, an attacker can execute arbitrary commands on the system or use other errors that may exist in the Bash command interpreter (if the attacker has a way to manipulate the list of environment variables and cause Bash to run).

The bug was released to the public on September 24, 2014, when Bash updated the patch and was ready to release, although it would take some time

- *Symantec Security Response*
  E-mail: see https://www.symantec.com/
- *Unix & Linux on stackexchange.*

pygain on CSDN

to update the computer to resolve potential security issues.

4 Analysis

The emergence of this script caused great concern to the technical staff, where *env* is a system command that causes the system to create an environment variable

x='() :;; echo vulnerable'

And with the value of this environment variable

bash -c "echo this is a test"

The "vulnerable" output from the first line exposes the existence of a vulnerability

Because the 'echo vulnerable' instruction following the function definition *'() :;;'* should not have been executed but was executed.

Detailed analysis of bash reveals that bash has design flaws in the code that handles the assignment of environment variables such as *"() :;;"* to function definitions, and incorrectly executes the strings following the function definition as commands.

So the real use has nothing to do with the env command, as long as you try to allow the system to accept an environment variable assignment with *"[function definition] + [arbitrary command]"* can trigger the code execution represented by the *"[arbitrary command]"* section.

5 Its Influence

For a system with a Bash vulnerability, because it allows unauthorized remote users to specify Bash environment variables, there are potential loopholes in the systems that run these services or applications.

Any program that can pass environment variables to bash by some means is affected by this. Of course, the most typical is the CGI program written by bash. The client can easily attack the server running CGI by adding the constructed value to the request string.

At present, most websites rarely use CGI, so the problem is not too great. However, many network devices, such as routers and switches, use CGI programs written in Perl or other languages. As long as the bottom layer calls bash, then there is a risk of love.

Any known program can be used to cause arbitrary command execution through a bash vulnerability as long as the following two conditions are met:

- The program uses bash as a script interpreter to handle environment variable assignment at a certain moment

- Submission of environment variable assignment string depends on user input

Currently, systems that may be used include:

- Apache HTTP Server running CGI scripts

- Linux-based routers that use CGI as a network interface

- Use Bash's various embedded devices

- Some DHCP clients

- Use Bash's various web services

- OpenSSH server using ForceCommand function

6 How Can It be Exploited?

While the vulnerability potentially affects any computer running Bash, it can only be exploited by a remote attacker in certain circumstances. For a successful attack to occur, an attacker needs to force an application to send a malicious environment variable to Bash.

The most likely route of attack is through Web servers utilizing CGI (Common Gateway Interface), the widely-used system for generating dynamic Web content. An attacker can potentially use CGI to send a malformed environment variable to a vulnerable Web server. Because the server uses Bash to interpret the variable, it will also run any malicious command tacked-on to it.

7 How to Fix the Bug?

The easiest way to fix the vulnerability is to use your default package manager to update the version of Bash.

on Debian

sudo apt update && sudo apt install –only-upgrade bash

on Fedora, Red Hat, Cent OS, etc.

yum -y update bash

8 The End

This is all about the Shellshock.

Created by IEEE Latex Template(bare_adv.tex actually). Most comments of the template were deleted.

BTW, because there are so many difficulties to use Chinese in LaTeX, I had to translate my original script into English.

Thanks!

References

[1] *Shellshock (software bug) - Wikipedia*

[2] *ShellShock: All you need to know about the Bash Bug vulnerability*

[3] *Trend Micro products and the Shellshock Linux Bash Vulnerability (Bash Bug) (CVE-2014-6271) and (CVE-2014-7169)*