

# Cryptography and Network Security

## Bash vulnerability - Shellshock

B00902029 余俊賢  
b00902029@ntu.edu.tw

B00902108 陳宣廷  
b00902108@ntu.edu.tw

B00902064 宋昊恩  
b00902064@ntu.edu.tw

B00902110 余孟桓  
b00902110@ntu.edu.tw

### ABSTRACT

在去年，繼 openssl 的 heartbleed 的漏洞後，又爆出了 bash 的 shellshock。當時看了很多科普的文章，都只不斷的重述該漏洞的嚴重性，但對於事件前後的細節、技術細節、成因等等，都不太提到。於是我們想對這個問題進行探究，了解這次事件的始末，以及來探討，身為未來可能會碰觸到各種 server 的資訊工程系學生，要怎麼樣保護好自己的服務，防止慘劇發生。

### Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous;  
D.2.8 [Software Engineering]: Metrics—*complexity measures, performance measures*

### General Terms

Cryptography

### Keywords

Shellshock, Bash Exploit

## 1. INTRODUCTION

### 1.1 Bash

Bash 是為了補強原有的 Bourne shell (一般人所知的 sh)，由 Brian J. Fox 為 GNU 計畫所寫，並在 1989 年發佈。而 bash 在下列平台上都是 default shell

- CentOS
- Fedora
- RHEL
- Mac OS X

而這些平台的使用者其實不少。例如，CentOS 常在企業中使用，Mac OS X 現今則是有不少擁戴者。

### 1.2 Environment variable

環境變數 (Environment variable)，每個 process 會有自己獨自的環境變數，在大多情況下，fork 出的 subprocess 會繼承 parent process 的環境變數。而環境變數主要的用途則是可以用輕易的個別設定每個 process，而不必去影響該 process 執行的 code。

某種程度上來說，該程式執行時可以透過偵測環境決定其行為，而環境變數則控制其環境。

## 2. WHAT IS SHELLSHOCK?

最一開始關於 bash 的漏洞，是 bash 有一個建立環境變數的語法，該語法的 parse 過程有問題，在插入某特定形式的環境變數時會異常的截斷，並把殘餘部分當作指令執行。

### 2.1 Anecdote

由於 bash 的 parse 自幹了 state machine，大量使用了 extern 開了不少全域變數。整體而言，整份 code 的修改跟 patch 會造的結果難以預期，而在後續的修補過程中也遭遇到 programmer 對於整個 parse 的流程沒有完全了解而導致 patch 不完整的例子。

欲了解更詳細資訊的話可以參考：

- CVE-2014-6277
- CVE-2014-6278
- CVE-2014-7169
- CVE-2014-7187

## 3. EXPLOIT

本質上，所有存在試圖呼叫 shell 且環境變數可以被控制的服務，都可以被利用。底下針對比較常見的幾種利用方式做介紹。

### 3.1 CGI-based web server

#### 3.1.1 Common Gateway Interface (CGI)

Common Gateway Interface (CGI)，CGI 是一個早期為了方便動態網頁設立的標準。其建立了 web server 與負責產生動態網頁內容的程之間的接口，使得負責產生網頁內容的程式可以被獨立挪出 web server 之外，並且可以透過 *stdio* 以及 環境變數與 web server 溝通。

#### 3.1.2 exploit in detail

其中，web server 會把 client 的資訊放在環境變數一起傳給負責產生內容的程式。所以，當該程式是以 bash 撰寫，或是執行過程有呼叫 bash 時，將會導致環境變數被 bash 繼承。然而，web server

傳遞的資訊包含了 HTTP Header，而且很不幸的 HTTP Header 是 client 存在於發起的 request 中，因此理所當然的是可控的。

基本上這類的攻擊會是拿到跑 web server 之受限的 user 的權限。一般而言可以讀取一些公開的敏感檔案 (ex. /etc/passwd)，或是當作 botnet 的一個 client。

但如果該台機器的檔案權限沒有設定好，導致可以任意寫入既有的檔案，或是用 root 權限跑起 web server，或是有一些本地提權的 kernel 漏洞，那就整台人生了。

### 3.1.3 example

首先，我們以最常見的 Apache 這個 web server 為例子。Apache 有很多 module，其中 mod\_cgi 與 mod\_cgid 處理了標準的 CGI 接口。在稍後我們將重現這個 service 以及其攻擊過程。

另外，mod\_fastcgi 與 mod\_cgid 處理了後來改版的 FastCGI 接口，也有一樣的問題。還好，後來 CGI 發展到各個網頁語言所衍生的 mod\_php, mod\_perl, mod\_python 以及 mod\_ruby 大多使用了一些保留的全域變數來放置 client 資訊，才沒有擴大災情。而實際上後期衍伸的接口也相對比較多人使用。

更詳細可以參考：

<http://timhsu.chroot.org/2014/10/bash-shellshock.html>

## 3.2 DHCP

### 3.2.1 Dynamic Host Configuration Protocol (DHCP)

Dynamic Host Configuration Protocol (DHCP)，顧名思義是個定義如何在內部網路動態由一個 server 配置 client 的網路資訊 (ex. IP) 的服務。

### 3.2.2 exploit

在各 linux 的 DHCP client 會把 DHCP server 給的一些資訊放入環境變數裡 (ex. 一些 option)。

當初測試可以攻陷 linux 系統有

- CentOS 7.0
- Debian 7.6
- Fedora 20
- Ubuntu 10.04.1 LTS
- Ubuntu 14.04.1 LTS

而這次的攻擊點可怕的是 DHCP client 通常是由 root 跑起來的，可以直接獲得系統的最高權限。但是 DHCP 僅限攻擊內網，而且如果 switch 本身有對 DHCP 封包做限制，僅能由受信任的 DHCP server 廣播的話，所能影響有限。

## 3.3 Limited OpenSSH servers

### 3.3.1 SSH Tunnel

在一些想要將 server 與 client 的連線加密的例子，會使用 ssh 實作，但是該實作會讓 server 在 client 正常 login 後只執行指定的程式，並將使用者關在該服務裡，避免其存取 shell，

例如，ptt, ptt2 的 ssh 加密連線，svn 與 git server 透過 ssh 連線。其中都在 server 的 sshd\_config 設定檔中 ForceCommand 設定好一旦 login 成功後要執行什麼，或是在 authorized\_keys 中利用 command= 設定之。

### 3.3.2 exploit

但是在 sshd 叫起該指定的指令的時候，會通過 login，並叫起 login shell，再跑起該指令，而 login shell 在上述的不少平台上，預設是 bash。

然而，環境變數是怎麼傳遞進去的呢？悲劇的事情是 ssh client 原本可以指定在 login 後要執行什麼指令，而該參數就算傳給 ssh server 被忽略掉不執行，也會被放進 SSH\_ORIGINAL\_COMMAND 這個環境變數裡。也就是說，透過 ssh user@pwnme.server 'payload' 而 payload 這個字串就會被放進環境變數裡。

這樣的攻擊可以跳脫原本的限制拿到該為了 ssh 加密通道而生的 user，基本上也是沒有什麼能炸掉整台 server 的動作，僅能存取公開的敏感檔案或是利用本地提權拿到 root。

## 4. DEMONSTRATION

### 4.1 Preparation

我們用 CGI 架了個包含 shellshock 漏洞的簡單網站。

注意到我們必須自行去 git clone 有漏洞的 bash 版本，因為當今的穩定版皆已修正此問題。另外由於我們架在 ubuntu 下，必須將預設 shell 由 dash 改為 bash。

此網站中我們使用了 Popen 去呼叫 date 來取得日期資訊，但以此就已經足夠重現此漏洞。

```
#####
# show a navbar with time
#####

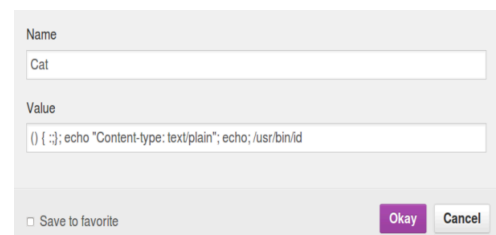
print ' <nav class="navbar navbar-default navbar-fixed-bottom">'
print ' <div class="container">'
try:
    p = Popen('date', shell=True, stdout=PIPE)
    res = p.communicate()[0]
    print 'Cute Date:', res
except Exception as e:
    print e
    print 'Current Time Error.'

print ' </div>'
print ' </nav>'
```

Figure 1:

### 4.2 Attack

我們使用 RESTClient 這個 Firefox 插件，在發送的 Header 中加入如下資料，會使伺服器嘗試執行 /usr/bin/id 指令



The screenshot shows the RESTClient plugin interface. The 'Name' field contains 'Cat'. The 'Value' field contains '() { :; }; echo "Content-type: text/plain"; echo; /usr/bin/id'. There are 'Save to favorite', 'Okay', and 'Cancel' buttons at the bottom.

Figure 2:

在原始碼中我們會看到 /usr/bin/id 已經被執行了，表示我們攻擊成功。一旦得到 shell，就有可能更進一步得到其他權限了。

## 5. CONCLUSION

當這個漏洞被發布的時候，patch 已經釋出了，因此其實對於有內建自動更新的機器影響不大。但是業界對於更新的習慣實在不是很

```

</head>
<body>
<script src="../../js/bootstrap.min.js"></script>
<nav class="navbar navbar-default navbar-fixed-bottom">
  <div class="container">
Cute Date: Content-type: text/plain

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

  </div>
</nav>
<div class="jumbotron">
  <div class="container">
    <h1>Cute Cats!</h1>

```

**Figure 3:**

好，或是常常使用已經不再維護的 OS 版本，更常見的損害則出現在一些硬體設備的嵌入式系統上，該系統大多都是廠商自己修改的作業系統且沒啟用自動更新，但是又長時間暴露在外網上。例如，router, switch, web cam ... 等等族繁不及備載。

身為未來很有可能進入業界工作的資訊工程學系的學生，對於怎麼樣保護自己與公司的機器，我想至少要有 定期更新這種常識等級的安全概念。

## 6. REFERENCE

- [1]: Bash code injection vulnerability (CVE-2014-6271, CVE-2014-7169),  
<https://access.redhat.com/node/1200223>
- [2]: DEV CORE blogs,  
<http://devco.re/blog/2014/09/30/shellshock-CVE-2014-6271/>
- [3]: Errata Security blogs,  
<http://blog.erratasec.com/2014/09/the-shockingly-bad-code-of-bash.html#.VX1IPxOqqko>
- [4]: CGI introduction,  
<http://ind.ntou.edu.tw/~dada/cgi/CGIintro.htm>
- [5]: Basic CGI concept,  
<http://mbsbears.com/teched/webdesign/kss/webdesign12/Webengineering/lec06.htm>
- [6]: Apache mod\_cgi implementation in python,  
<https://www.exploit-db.com/exploits/34900/>
- [7]: Inside Shellshock: How hackers are using it to exploit systems,  
<https://blog.cloudflare.com/inside-shellshock/>