可靠可重复地构建 Debian 软件包

Boyuan Yang

USTC Linux User Group

2016 年 11 月 24 日

内容概览

Debian 软件现状

为了安全，已经做到了什么？

作为开源软件项目，Debian 提供的软件有以下的特性保证安全：

- 使用 GPG 签名验证源代码可信与否（需要上游支持）；
- 提供编译选项进行安全增强；
- 提供经过签名的源码包供用户自行编译使用；
- 对二进制包所在软件仓库[1]签名确保安全性；
- 部分镜像站点启用 HTTPS 确保传输安全。

[1]默认情况下并没有对单独的二进制.deb 包签名，这是一个问题。

Debian 软件现状

为了安全，已经做到了什么？

作为开源软件项目，Debian 提供的软件有以下的特性保证安全：

- 使用 GPG 签名验证源代码可信与否（需要上游支持）；
- 提供编译选项进行安全增强；
- 提供经过签名的源码包供用户自行编译使用；
- 对二进制包所在软件仓库[1]签名确保安全性；
- 部分镜像站点启用 HTTPS 确保传输安全。

存在一个盲点：无法确保从源代码到所构建二进制软件包的对应关系与安全性。

[1]默认情况下并没有对单独的二进制.deb 包签名，这是一个问题。

Debian 软件现状

无法保证构建稳定性，为什么不安全？

构建过程不确定，就无法确保二进制软件包的安全性。

1. Debian 开发者可能为了某种利益[2]故意上传带后门的二进制 deb 包；
2. 攻击实例：CVE-2002-0083 for OpenSSH，单比特翻转导致远程提权漏洞；
3. 编译器自身的问题，如 XCodeGhost。

[2] Debian 官方直到 2014 年才开始接受仅包含源代码的上传（SourceOnlyUpload）仅上传源代码可以阻止开发者有意准备含有后门或漏洞的二进制软件包。直到今天，Debian 仓库仍然允许二进制包的上传，且在可预见的未来将一直如此。[2]

Debian 软件现状

无法保证构建稳定性，为什么不安全？

构建过程不确定，就无法确保二进制软件包的安全性。

1. Debian 开发者可能为了某种利益[2]故意上传带后门的二进制 deb 包；
2. 攻击实例：CVE-2002-0083 for OpenSSH，单比特翻转导致远程提权漏洞；
3. 编译器自身的问题，如 XCodeGhost。

我们希望构建过程是稳定的，是可重现的。

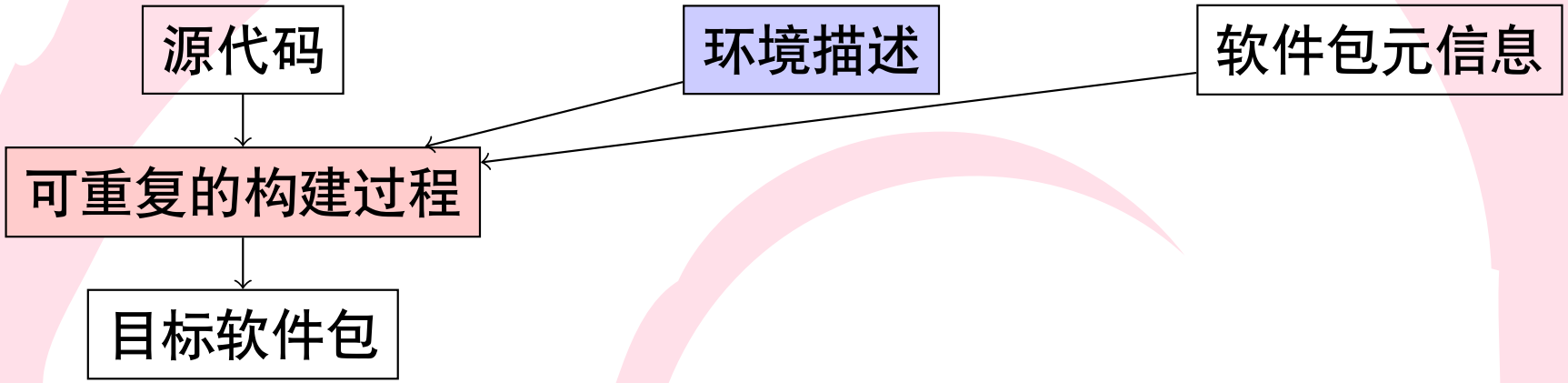
> *The motivation behind "reproducible" builds is to allow verification that no flaws have been introduced during the compilation process.[4]*

[2] Debian 官方直到 2014 年才开始接受仅包含源代码的上传（SourceOnlyUpload）仅上传源代码可以阻止开发者有意准备含有后门或漏洞的二进制软件包。直到今天，Debian 仓库仍然允许二进制包的上传，且在可预见的未来将一直如此。[2]

工作目标

期望效果与解决方案

期望达到的目标：确定的条件下构建的软件包完全相同（“bit-to-bit identical”）。



于是，我们可以在多个不同的场合下对软件进行重新构建，给出的二进制包与构建所得的二进制包互相对照是否相同以确保给出的二进制包与源码包的对应关系。

可重复构建面临的挑战

不确定性因素的分类

要排除不确定性因素，包括以下各个方面：

1. 时间戳；
2. 时区与地区格式；
3. 不确定的文件排序；
4. 不确定的字典、散列值排序；
5. 用户、用户组、umask、环境变量；
6. 构建路径（build path）；[3]
7. 其它与环境相关的可变情况。

[3]在 Debian 9 中容忍同样的构建路径，在 sid 中则将其纳入考虑范围内。

Debian 中开展可重复构建工作

工作思路

- 在某个 Debian 环境中连续两次构建一个软件包，改变如下变量：
  1. 时间、日期；
  2. 主机名、域名；
  3. 文件系统（disorderfs?）；
  4. 时区、地区 locale；
  5. 用户 ID、用户组 ID；
  6. 内核版本、CPU 厂商；
  7. /bin/sh 默认 shell；
  8. 各类其它环境变量，等等。
- 比较两者构建结果是否相同：
  - 如相同，记录结果；
  - 如不同，使用特定工具（diffoscope）检查构建结果，寻找不同点并找到原因所在，记录并提交缺陷报告；
  - 与上游一起工作，提交补丁，等等。

可重复构建需要的工具

通用工具

diffoscope：深入压缩包与文件的 diff 增强版

由 reproducible-builds 团队领衔开发，专门用来探查 .deb 包之间的差异究竟在哪里，便于对症下药。

data.tar.xz

data.tar

./usr/share/doc/qevercloud-doc/qevercloud.pdf

pdftk {} output - uncompress

Offset 254616, 14 lines modified	Offset 254616, 14 lines modified
2546160004955227 00000 n	2546160004955227 00000 n
2546170004955271 00000 n	2546170004955271 00000 n
2546180004955480 00000 n	2546180004955480 00000 n
254619trailer	254619trailer
254620<	254620<
254621/Info 16799 0 R	254621/Info 16799 0 R
254622/ID [<472d0ddfb78d871ecec2d7d4d3599476> <472d0ddfb78d871ecec2d7d4d3599476>]	254622/ID [<f5b6de527f01e0d942a6f34b410f6961> <f5b6de527f01e0d942a6f34b410f6961>]
254623/Root 16798 0 R	254623/Root 16798 0 R
254624/Size 16800	254624/Size 16800

dpkg：工具链核心，构建稳定的 .deb 包

所有项目需要的补丁在 dpkg 1.18.11 版并入主线，从 Debian 9 或 Ubuntu 17.04 版开始默认可使用。

另外项目团队还有一些正在开发的工具（如 debrebuild、debrepatch、reprotest 等，起到方便对软件包是否可重复构建进行检查等作用。

.buildinfo 文件：对构建环境进行描述

它给出了构建时的输入与输出。

- 输入：.dsc 文件、构建依赖及其版本、构建路径等；
- 输出：.deb 校验和等。

临时存储服务器：https://buildinfo.debian.net

构建效果展示

阶段性成果展示

我们可以重复构建一个软件包，看看结果是否相同。

1. 比较散列值即可；
2. 如果受到时间戳等因素的影响，任何两次构建都不会是相同的。

构建效果展示

阶段性成果展示

我们可以重复构建一个软件包，看看结果是否相同。

1. 比较散列值即可；
2. 如果受到时间戳等因素的影响，任何两次构建都不会是相同的。

请看演示：

SOURCE_DATE_EPOCH：~~独立自主~~制定的环境变量标准

- 纯数字的 UNIX 时间戳；
- 如果环境变量中给定了这个变量，则将所有自己程序的时间戳设为该变量对应的时间。[3]
- 已经被包括 gcc 在内的各大项目所接受，从根本上解决了时间戳问题。

工作成果展示

与上游的合作

团队向各种上游发了各式各样的缺陷报告与补丁。略举数例：

- gcc 的 `__DATE__` 与 `__TIME__` 宏采用 `SOURCE_DATE_EPOCH` 来防止时间戳问题；
- python 3.6 开始提供有序字典类型；[4]
- TeXLive 2016 开始，除了 LuaTeX 和原始 TeX 以外均支持 `SOURCE_DATE_EPOCH`；[5]
- doxygen 时间戳问题。[6]

[4] https://mail.python.org/pipermail/python-dev/2016-September/146327.html

[5] https://www.preining.info/blog/2016/06/tex-live-2016-released/

[6] https://bugs.debian.org/792201

工作进度展示

Debian 9(Stretch) 完成情况

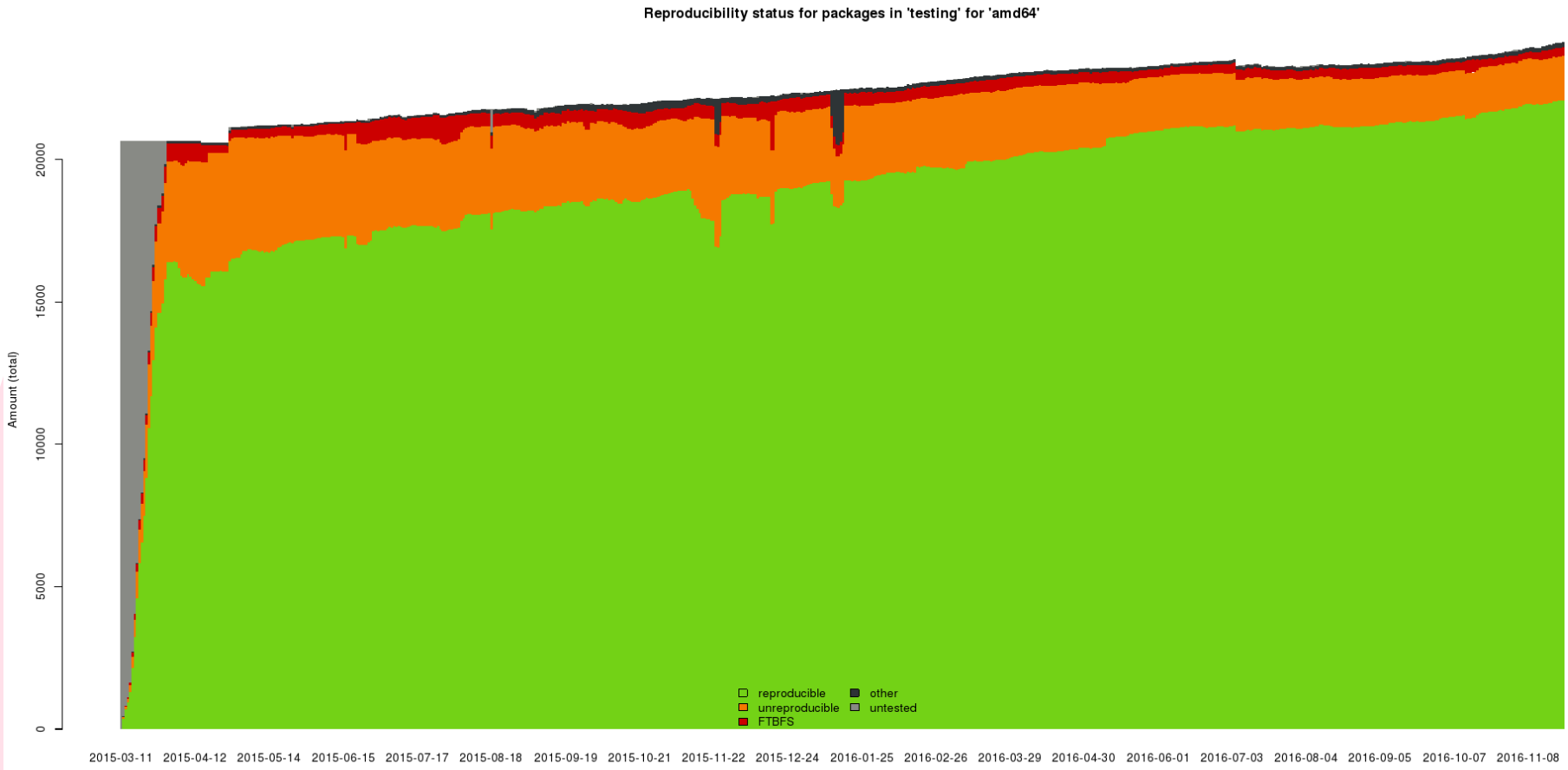
即将发布的 Debian 9 可以保证 90% 以上的软件包可靠地重复构建。[1]

套件	all	reproducible	unrep	FTBFS
testing/amd64	24132	22090 / 91.5%	1557 / 6.4%	291 / 1.2%
testing/i386	24132	21355 / 88.4%	2170 / 8.9%	374 / 1.5%
testing/armhf	24132	21009 / 87.1%	1852 / 7.6%	695 / 2.8%
sid/amd64	25399	19182 / 75.5%	4501 / 17.7%	944 / 3.7%
sid/i386	25399	18632 / 73.3%	5054 / 19.8%	1153 / 4.5%
sid/armhf	25399	18591 / 73.2%	4823 / 18.9%	1215 / 4.7%

工作进度展示

Debian 9(Stretch) 完成情况

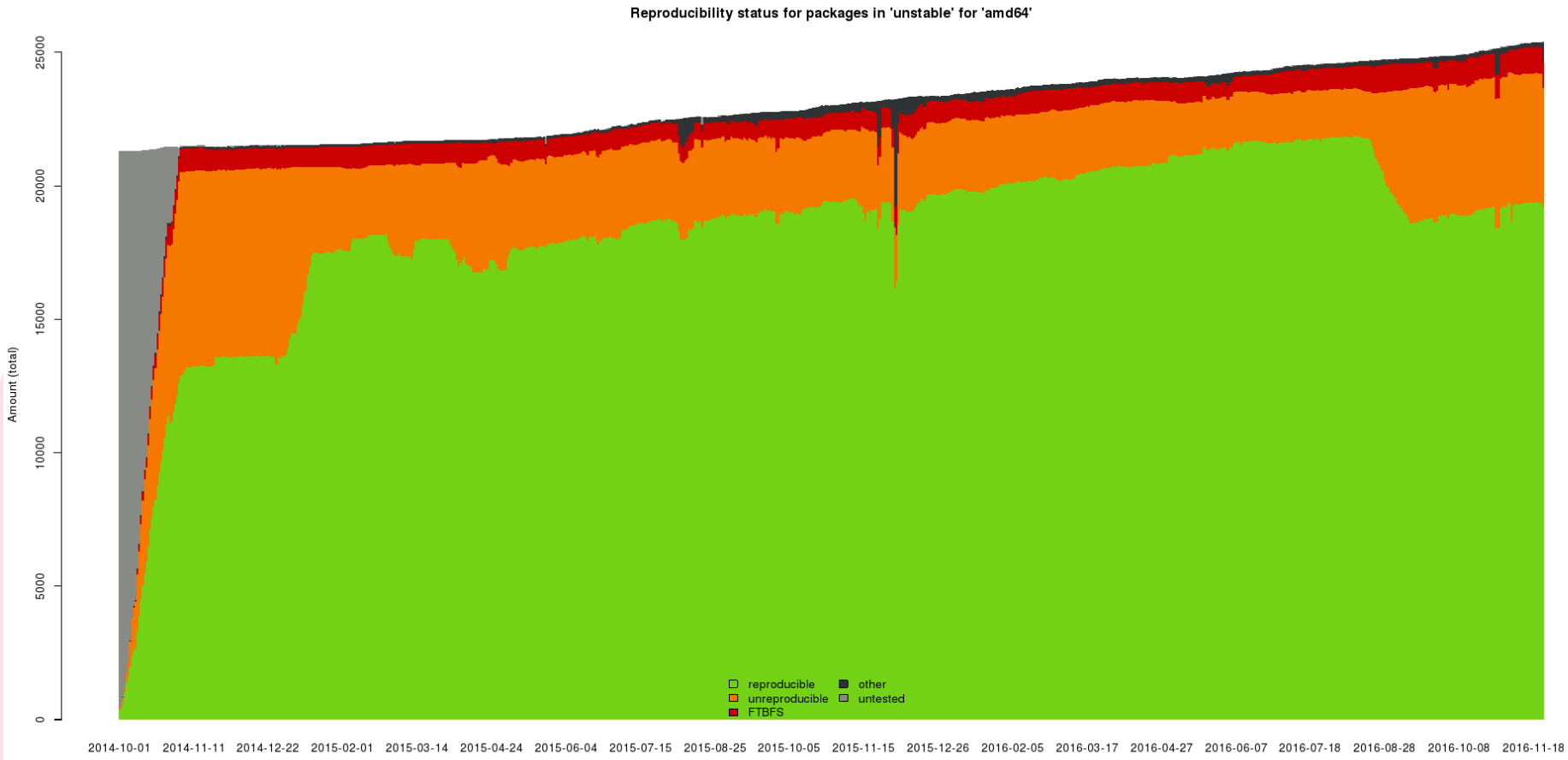
Debian Testing (9) x86_64 平台测试情况



工作进度展示

Debian 9(Stretch) 完成情况

Debian Unstable (sid) x86_64 平台测试情况



其他参与可重复性构建的软件项目

- Arch Linux
- Bitcoin
- coreboot
- Debian
- F-Droid
- Fedora
- FreeBSD
- Guix
- LEDE
- NetBSD
- OpenWrt
- openSUSE
- Tails
- Tor Browser

……还有更多!

未来展望

项目带来的额外优势

使用持续集成方式探测检查软件包的可重复构建性，还可能带来以下额外好处：

1. 将 buildd 的一部分构建任务交给持续集成系统，减轻负担；
2. 承担一部分质保任务，探测软件包 regression；
3. 检查出软件在边界条件上可能出现的缺陷；
4. 避免不确定性因素对软件质量的影响（互联网质量、时间戳等）。

参考

项目资源

- 官方网站：https://reproducible-builds.org
- Debian 测试情况：https://reproducible.debian.net
- Debian Wiki 主页：
  https://wiki.debian.org/ReproducibleBuilds
- 项目每周报告：
  https://reproducible.alioth.debian.org/blog/
- 项目历史记录：
  https://wiki.debian.org/ReproducibleBuilds/History

参考

进一步阅读

- the Debian Project. *Overview of various statistics about reproducible builds*. 2016. URL: https://tests.reproducible-builds.org/debian/reproducible.html (visited on 11/24/2016).
- the Debian Project. *Source Only Upload*. 2016. URL: https://wiki.debian.org/SourceOnlyUpload (visited on 11/23/2016).
- Chris Lamb. *SOURCE_DATE_EPOCH specification*. 2015. URL: https://reproducible-builds.org/specs/source-date-epoch/.
- Chris Lamb and Holger Levsen. *Reproducible builds status update*. 2016. URL: https://people.debian.org/~lamby/2016-11-13-MiniDebConfCambridge/ (visited on 11/23/2016).

问 答 环 节

版权声明
Copyright Notice