Introduction to Reproducible Builds

Vagrant Cascadian

Scale15x 2017-03-05
The Reproducible Builds project aims to bring us closer to a world where binary software can be independently verified as the result of building the provided source code.
Source Code

- Source code is readable and writeable by trained monkeys humans
- Computers run binary code
- How do you know the binary code the computer is running was produced from the source code?
Reproducibility is the ability of an entire experiment or study to be duplicated, either by the same researcher or by someone else working independently.

Oooh, Math(s)!

```python
c $ python -c 'x=1 ; y=1 ; print(x+y)'  
2

c $ python -c 'x=1 ; y=1 ; print(x+y)' | sha256sum  
53c234e5e8472b6ac...8977b010655bfdd3c3  -

c $ echo 2 | sha256sum  
53c234e5e8472b6ac...8977b010655bfdd3c3  -
```
But software building is more like...

\[ x = \text{source code} \]
\[ y = \text{build arguments} \]
\[ z = \text{toolchain (compiler, liker, libraries, etc.)} \]
\[ r = \text{other stuff (time of build, running OS, username building software, environment variables, etc.)} \]
\[ x + y + z + r = ? \]
Independent verification

source code + build environment + build instructions
=
bit-by-bit identical copies
anyone can verify the result
https://reproducible-builds.org/docs/definition/
History in Debian

- Mentioned on lists as early as 2007
- Didn’t gain traction until more recently
- Automated rebuilding of Debian’s 25,000+ source packages began in late 2014
- Currently rebuilding roughly 1,600-2,200 packages a day on each of amd64, i386, arm64 and armhf
A plague of unreproducibility

- About 4,800 (19%) of software in Debian unstable
- About 1,300 (5%) of software in Debian testing
- Patches in Debian toolchains and packages, but patches are swimming upstream
What kind of security implications are we facing?

- **CVE-2002-0083**: Remote root exploit in OpenSSH, caused by an off-by-one error
- **2015: XcodeGhost**: Malware variant of Apple’s SDK Infected over 4,000 apps in Apple’s App store
T rusting T rust

Reflections on Trusting Trust by Ken Thompson 1984

- https://www.ece.cmu.edu/~ganger/712.fall02/papers/p761-thompson.pdf
Diverse Double-Compilation by David A. Wheeler 2005/2009

- https://www.dwheeler.com/trusting-trust/
Common problems

- timestamps
- timezone
- file sort order
- locales
Embedded timestamps:

U-Boot SPL 2016.01+dfsg1-3 (Feb 21 2016 - 21:39:10)
There’s no timestamps like NO timestamps.
If you really must, use the SOURCE_DATE_EPOCH specification, which specifies the timestamp to use in a standardized environment variable.

https://reproducible-builds.org/specs/source-date-epoch/
The timezone of the running build can impact output:

$ LC_ALL=C date –date "@1478647393" –rfc-2822 Tue, 08 Nov 2016 15:23:13 -0800

Set to UTC using TZ environment variable:

$ TZ=UTC LC_ALL=C date –date "@1478647393" –rfc-2822 Tue, 08 Nov 2016 23:23:13 +0000

https://reproducible-builds.org/docs/timezones/
file sort order

- Bad Makefile:

SRCS = $(wildcard *.c)
tool: $(SRCS:.c=.o)
   $(CC) -o $@ $^

- Good Makefile:

SRCS = $(sort $(wildcard *.c))
tool: $(SRCS:.c=.o)
   $(CC) -o $@ $^

https://reproducible-builds.org/docs/stable-inputs/
Sort order for C, as spoken in UNIX:

$ printf 'a\nB\nb\nA\n' | LC_ALL=C sort
  A
  B
  a
  b

Sort order for English, as spoken in USA:

$ printf 'a\nB\nb\nA\n' | LC_ALL=en_US.UTF-8 sort
  a
  A
  b
  B

https://reproducible-builds.org/docs/locales/
Build path

- Hard to fix correctly
- Ongoing work to GCC and other major toolchains by Ximin Luo and others:
  - Some patches to GCC accepted, more in progress
  - draft specification: BUILD_PATH_PREFIX_MAP in progress
- Can be worked around by normalized build environment
Write your code with intention

- Remove unintended inputs
- Remove random inputs
- Verifyable built results
- Gain confidence in your builds
Buildinfo files

- specification:
  https://manpages.debian.org/jump?q=deb-buildinfo

- examples from the real world:
  https://buildinfo.debian.net
Example .buildinfo

Source: libtext-simpletable-perl
Version: 2.03-1
Checksums-Sha256:
    7a285...a8b 10788 libtext-simpletable-perl_2.03-1_all.deb
Build-Architecture: amd64
Build-Date: Fri, 03 Mar 2017 07:56:17 +1400
Build-Path: /build/libtext-simpletable-perl-2.03/2nd
Installed-Build-Depends:
    autoconf (= 2.69-10),
    automake (= 1:1.15-6),
    zlib1g (= 1:1.2.8.dfsg-5)
Environment:
    DEB_BUILD_OPTIONS="parallel=15"
    LANG="C"
    LC_ALL="C"
    SOURCE_DATE_EPOCH="1439466701"
Building tools

- reprotest - source rebuilder

  reprotest 'dpkg-buildpackage -b --no-sign' ' ../*.deb'

- debrepro - simple .deb rebuilder

  debrepro
diffoscope - an exceptionally clever diff tool
https://diffoscope.org
diff as a service:
https://try.diffoscope.org/
trydiffoscope client
Other projects

It goes well beyond Debian:
https://reproducible-builds.org/who/

- NixOS
- GNU Guix
- Fedora
- OpenSUSE
- FreeBSD
- Arch Linux
- Tails
- Bitcoin
- Coreboot
- Tor Browser
- And more...
Thanks

- Core Infrastructure Initiative
- Profitbricks
- Codethink

All the great folks doing reproducible builds work!
Copyright 2016-2017 Vagrant Cascadian <vagrant@debian.org>
Copyright of images included in this document are held by their respective owners.
This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.
To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/4.0/