Introduction to Reproducible Builds

Vagrant Cascadian

SeaGL 2016-11-12

Goals

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?

Scientific Methods

Reproducibility is the ability of an entire experiment or study to be duplicated, either by the same researcher or by someone else working independently.

`https://en.wikipedia.org/wiki/Reproducibility`

Ooooh, Math(s)!

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

$ python -c 'x=1 ; y=1 ; print(x+y)' | md5sum
26ab0db90d72e28ad0ba1e22ee510510  -

$ echo 2 | md5sum
26ab0db90d72e28ad0ba1e22ee510510  -
```

But software building is more like. . .

x=1
y=1
z=toolchain (compiler, liker, libraries, etc.)
r=other stuff (time of build, running OS, username building software, etc.)
$x + y + z + r = ?$

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 1,800 packages a day on each of amd64, i386 and armhf

A plague of unreproducibility

Recent status with magic numbers:

- About 4,800 (19%) of software in Debian unstable
- About 1,600 (7%) of software in Debian testing
- Debian unstable has more things varied between builds
- Patches in Debian toolchains and packages, but patches are swimming upstream

Reproducibility matters

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

Common problems

- timestamps
- timezone
- file sort order
- locales

timestamps

- Embedded timestamps:

U-Boot SPL 2016.01+dfsg1-3 (Feb 21 2016 - 21:39:10)

timestamps: Please No

- There's no timestamps like NO timestamps.

timestamps: SOURCE_DATE_EPOCH

- 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/`

timezone

- 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/

locales

- 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/

Building tools

- reprotest - source rebuilder

```
reprotest 'debuild -b -uc -us' '../*.deb'
```

- debrepro - simple .deb rebuilder

```
debrepro
```

diffoscope

diffoscope - an exceptionally clever diff tool
`https://diffoscope.org`

try.diffoscope.org

- diff as a service: `https://try.diffoscope.org/`
- trydiffoscope client

Thanks

Profitbricks
Core Infrastructure Initiative
Lunar
Holger Levsen
Chris Lamb
Reiner Herrmann
All the other great folks doing reproducible builds work!

Copyright