Reproducible builds everywhere

Bit by bit identical binaries
from a given source

Mattia Rizzolo

Debian-Ubuntu Community Conference — Italia 2017
Vicenza, Italy
2017-05-06

about Mattia

- `66AE 2B4A FCCF 3F52 DA18 4D18 4B04 3FCD B944 4540`
- Ubuntu/Debian contributor since 2010
- Ubuntu Developer since the last December
- Debian Developer since the second last December
  - Debian QA (quality assurance)
    - MIA Team
    - `https://jenkins.debian.net` (1000 jobs continously testing Debian)
    - ...
  - Debian Reproducible builds team member
    - since August 2016 funded by the Linux Foundation
  - Quite a bunch of other stuff ...

Debian reproducible builds contributors

akira
Alexis Bienvenüe
Andrew Ayer
Asheesh Laroia
Boyuan Yang
Ceridwen
Chris Lamb
Chris West
Christoph Berg
Clint Adams
Dafydd Harries
Daniel Kahn Gillmor
Daniel Shahaf
Daniel Stender
David Suarez
Dhole
Drew Fisher
Emmanuel Bourg
Emanuel Bronshtein
Esa Peuha
Fabian Wolff
Guillem Jover
Hans-Christoph Steiner
Harlan Lieberman-Berg
Helmut Grohne
Holger Levsen
HW42
Intrigeri
Jelmer Vernooij
josch
Juan Picca
Lunar
Maria Glukhova
Mathieu Bridon
Mattia Rizzolo
Nicolas Boulenguez
Niels Thykier
Niko Tyni
Paul Wise
Peter De Wachter
Philip Rinn
Reiner Herrmann
Robbie Harwood
Santiago Vila
Sascha Steinbiss
Satyam Zode
Scarlett Clark
Stefano Rivera
Stéphane Glondu
Steven Chamberlain
Tom Fitzhenry
Valerie Young
Valentin Lorentz
Wookey
Ximin Luo

Debian reproducible builds contributors

akira
Alexis Bienvenüe
Andrew Ayer
Asheesh Laroia
Boyuan Yang
Ceridwen
Chris Lamb
Chris West
Christoph Berg
Clint Adams
Dafydd Harries
Daniel Kahn Gillmor
Daniel Shahaf
Daniel Stender
David Suarez
Dhole
Drew Fisher
Emmanuel Bourg
Emanuel Bronshtein
Esa Peuha
Fabian Wolff
Guillem Jover
Hans-Christoph Steiner
Harlan Lieberman-Berg
Helmut Grohne
Holger Levsen
HW42
Intrigeri
Jelmer Vernooij
josch
Juan Picca
Lunar
Maria Glukhova
Mathieu Bridon
Mattia Rizzolo
Nicolas Boulenguez
Niels Thykier
Niko Tyni
Paul Wise
Peter De Wachter
Philip Rinn
Reiner Herrmann
Robbie Harwood
Santiago Vila
Sascha Steinbiss
Satyam Zode
Scarlett Clark
Stefano Rivera
Stéphane Glondu
Steven Chamberlain
Tom Fitzhenry
Valerie Young
Valentin Lorentz
Wookey
Ximin Luo

Who are you?

Who are you?

- Seen a talk about reproducible builds?

Who are you?

- Seen a talk about reproducible builds?
- Contributed to the effort?

Who are you?

- Seen a talk about reproducible builds?
- Contributed to the effort?
- Uses Debian or a Debian based systems?

Who are you?

- Seen a talk about reproducible builds?
- Contributed to the effort?
- Uses Debian or a Debian based systems?
- Uses Fedora, RHEL, CentOS or a Fedora derivative based systems?

Who are you?

- Seen a talk about reproducible builds?
- Contributed to the effort?
- Uses Debian or a Debian based systems?
- Uses Fedora, RHEL, CentOS or a Fedora derivative based systems?
- BSD?

The problem: we need to believe

- Free Software is great: one can study, modify, share and use it!

The problem: we need to believe

- Free Software is great: one can study, modify, share and use it!
- We study, modify and share source code.
- We use binaries.

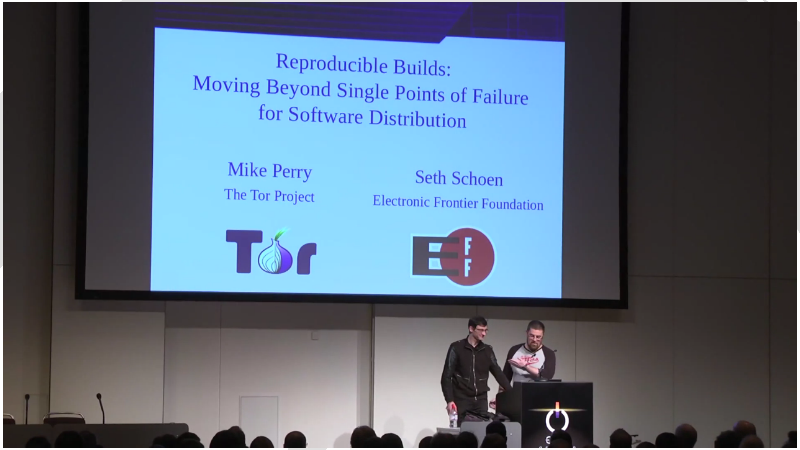
The problem: we need to believe

- Free Software is great: one can study, modify, share and use it!
- We study, modify and share source code.
- We use binaries.
- We need to believe our binaries come from the source code they are said to made from.

The problem: we need to believe

- Free Software is great: one can study, modify, share and use it!
- We study, modify and share source code.
- We use binaries.
- We need to believe our binaries come from the source code they are said to made from.
- **I do not want to believe.**

The problem in greater detail


Reproducible Builds:
Moving Beyond Single Points of Failure
for Software Distribution

Mike Perry
The Tor Project

Seth Schoen
Electronic Frontier Foundation

Available on `media.ccc.de`, 31c3

A few examples from that 31c3 talk

- CVE-2002-0083: remote root exploit in `sshd`, a single bit difference in the binary

A few examples from that 31c3 talk

- CVE-2002-0083: remote root exploit in `sshd`, a single bit difference in the binary
- 31c3 talk had a live demo with a kernel module modifying source code in memory only

A few examples from that 31c3 talk

- CVE-2002-0083: remote root exploit in `sshd`, a single bit difference in the binary
- 31c3 talk had a live demo with a kernel module modifying source code in memory only
- How can you be sure what's running on your machine or on a build daemon network connected to the net? Do you ever leave your computers physically alone?

A few examples from that 31c3 talk

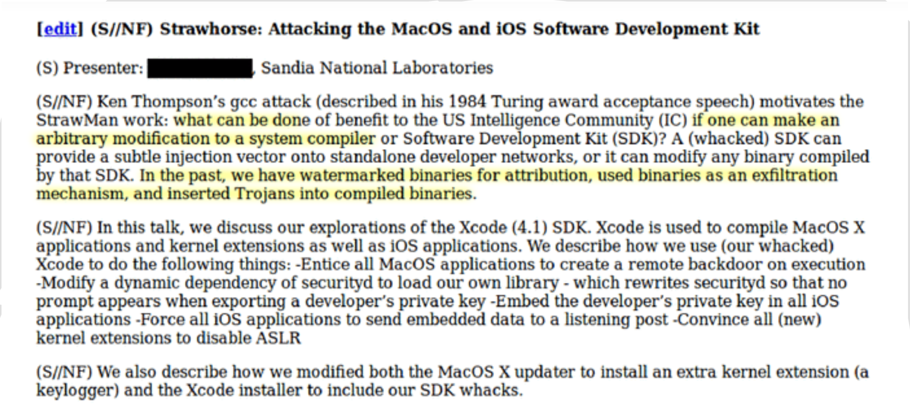
- CVE-2002-0083: remote root exploit in `sshd`, a single bit difference in the binary
- 31c3 talk had a live demo with a kernel module modifying source code in memory only
- How can you be sure what's running on your machine or on a build daemon network connected to the net? Do you ever leave your computers physically alone?
- How much do you pay your admins? Enough to withstand a multi million dollar attack?

A few examples from that 31c3 talk

- CVE-2002-0083: remote root exploit in `sshd`, a single bit difference in the binary
- 31c3 talk had a live demo with a kernel module modifying source code in memory only
- How can you be sure what's running on your machine or on a build daemon network connected to the net? Do you ever leave your computers physically alone?
- How much do you pay your admins? Enough to withstand a multi million dollar attack?
- Legal challenges. Could you be forced to backdoor (some of) your software (for some customers)?

Another example from real life

At a CIA conference in 2012:



[edit] (S//NF) Strawhorse: Attacking the MacOS and iOS Software Development Kit

(S) Presenter: [redacted], Sandia National Laboratories

(S//NF) Ken Thompson's gcc attack (described in his 1984 Turing award acceptance speech) motivates the StrawMan work: what can be done of benefit to the US Intelligence Community (IC) if one can make an arbitrary modification to a system compiler or Software Development Kit (SDK)? A (whacked) SDK can provide a subtle injection vector onto standalone developer networks, or it can modify any binary compiled by that SDK. In the past, we have watermarked binaries for attribution, used binaries as an exfiltration mechanism, and inserted Trojans into compiled binaries.

(S//NF) In this talk, we discuss our explorations of the Xcode (4.1) SDK. Xcode is used to compile MacOS X applications and kernel extensions as well as iOS applications. We describe how we use (our whacked) Xcode to do the following things: -Entice all MacOS applications to create a remote backdoor on execution -Modify a dynamic dependency of securityd to load our own library - which rewrites securityd so that no prompt appears when exporting a developer's private key -Embed the developer's private key in all iOS applications -Force all iOS applications to send embedded data to a listening post -Convince all (new) kernel extensions to disable ASLR

(S//NF) We also describe how we modified both the MacOS X updater to install an extra kernel extension (a keylogger) and the Xcode installer to include our SDK whacks.

`firstlook.org/theintercept/2015/03/10/ispy-cia-campaign-steal-apples-secrets/`

The solution

Promise that anyone can always and independently generate identical binary packages from a given source

The solution

We call this:

“Reproducible builds”

Debian demo

- Build a package 5 times, get 5 .debs with different checksums

Debian demo

- Build a package 5 times, get 5 .debs with different checksums
- Build a package 5 times, get 5 .debs with the same checksum

Debian demo

- Build a package 5 times, get 5 .debs with different checksums
- Build a package 5 times, get 5 .debs with the same checksum
- Yes, it's really this simple.

This should become the **norm**.

This should become the **norm**.

We want to change the meaning of ”free software”:

it’s only free software if it’s reproducible!

More benefits than ”just” security...

- Lots and lots of QA benefits - we’ve found so many subtile bugs.

More benefits than ”just” security...

- Lots and lots of QA benefits - we’ve found so many subtile bugs.
- Google does reproducible builds, to save time and money.

More benefits than ”just” security...

- Lots and lots of QA benefits - we’ve found so many subtile bugs.
- Google does reproducible builds, to save time and money.
- Smaller deltas, thus faster updates possible (for packages and images).

More benefits than ”just” security...

- Lots and lots of QA benefits - we’ve found so many subtile bugs.
- Google does reproducible builds, to save time and money.
- Smaller deltas, thus faster updates possible (for packages and images).
- Side effect: meaningful binary diff between two versions.

More benefits than "just" security...

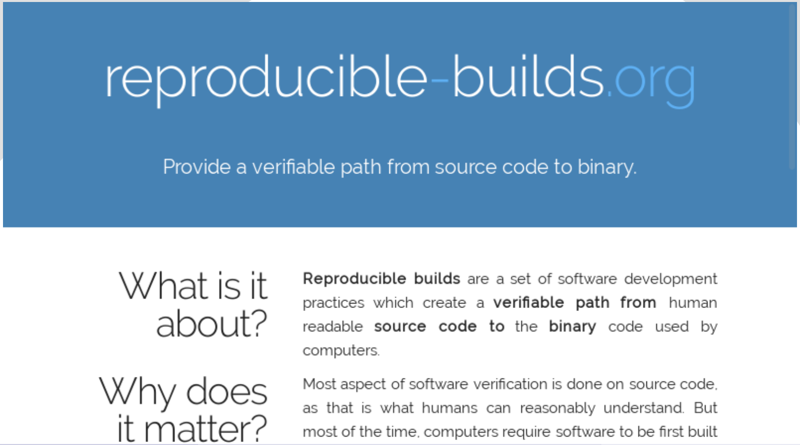
- Lots and lots of QA benefits - we've found so many subtile bugs.
- Google does reproducible builds, to save time and money.
- Smaller deltas, thus faster updates possible (for packages and images).
- Side effect: meaningful binary diff between two versions.
- ...

Disclaimer:

- these slides contain 5 week old data.
- ...

reproducible-builds.org

- `https://reproducible-builds.org`
- git repositories, IRC channels, mailinglists, webspace



reproducible-builds.org

Provide a verifiable path from source code to binary.

What is it about?

Reproducible builds are a set of software development practices which create a **verifiable path from** human readable **source code to** the **binary** code used by computers.

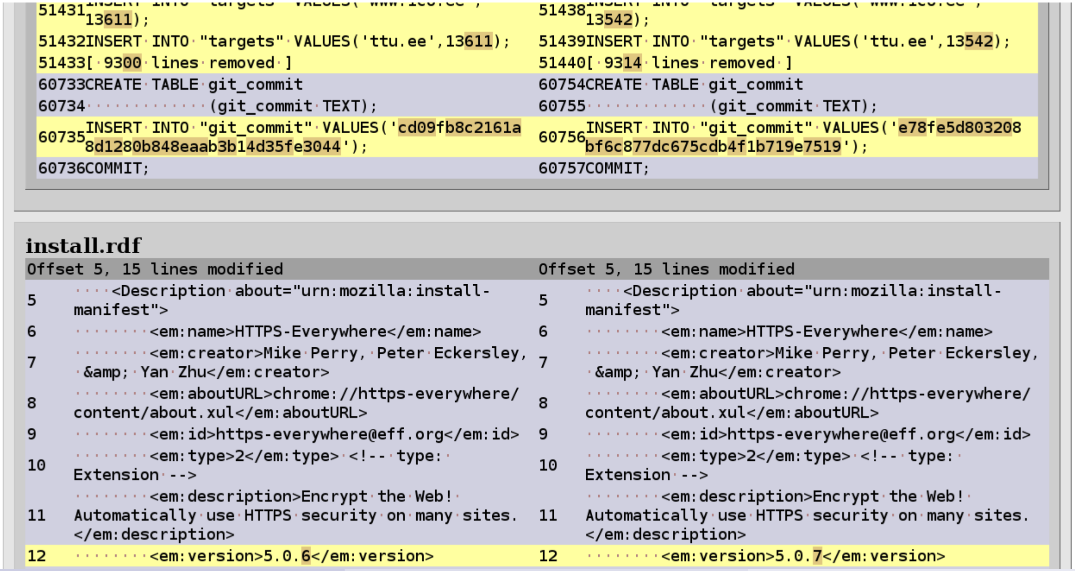
Why does it matter?

Most aspect of software verification is done on source code, as that is what humans can reasonably understand. But most of the time, computers require software to be first built

Debugging problems: `https://try.diffoscope.org`

- Examines differences **in depth**.
- Recursively unpacks archives, uncompresses PDFs, disassembles binaries, unpacks Gettext files, ...
- Easy to extend to new file formats.
- Falls back to binary comparison.
- Outputs HTML or plain text with human readable differences.
- Available from `git`, PyPI, Debian, Arch Linux, Guix, Homebrew, Fedora. Works on BSD.
- Maintainers in other distros wanted.
- `https://diffoscope.org/`

diffoscope example (HTML output)



```
51431 13611);                                               51438 13542);
51432INSERT INTO "targets" VALUES('ttu.ee',13611);          51439INSERT INTO "targets" VALUES('ttu.ee',13542);
51433[ 9300 lines removed ]                                 51440[ 9314 lines removed ]
60733CREATE TABLE git_commit                                60754CREATE TABLE git_commit
60734             (git_commit TEXT);                        60755             (git_commit TEXT);
60735INSERT INTO "git_commit" VALUES('cd09fb8c2161a           60756INSERT INTO "git_commit" VALUES('e78fe5d803208
     8d1280b848eaab3b14d35fe3044');                              bf6c877dc675cdb4f1b719e7519');
60736COMMIT;                                                 60757COMMIT;
```

install.rdf

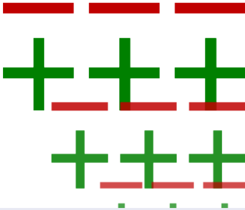
```
Offset 5, 15 lines modified                                 Offset 5, 15 lines modified
5     <Description about="urn:mozilla:install-               5     <Description about="urn:mozilla:install-
     manifest">                                                  manifest">
6         <em:name>HTTPS-Everywhere</em:name>               6         <em:name>HTTPS-Everywhere</em:name>
7         <em:creator>Mike Perry, Peter Eckersley,          7         <em:creator>Mike Perry, Peter Eckersley,
      &amp; Yan Zhu</em:creator>                                 &amp; Yan Zhu</em:creator>
8         <em:aboutURL>chrome://https-everywhere/           8         <em:aboutURL>chrome://https-everywhere/
     content/about.xul</em:aboutURL>                             content/about.xul</em:aboutURL>
9         <em:id>https-everywhere@eff.org</em:id>           9         <em:id>https-everywhere@eff.org</em:id>
10        <em:type>2</em:type> <!-- type:                   10        <em:type>2</em:type> <!-- type:
     Extension -->                                               Extension -->
          <em:description>Encrypt the Web!                          <em:description>Encrypt the Web!
11   Automatically use HTTPS security on many sites.        11   Automatically use HTTPS security on many sites.
     </em:description>                                           </em:description>
12        <em:version>5.0.6</em:version>                    12        <em:version>5.0.7</em:version>
```

`diffoscope` is "just" for debugging

- Reminder: `diffoscope` is for **debugging**
- "reproducible" according to our definition means: **bit by bit identical**. So the tools for testing whether something is reproducible are either `diff` or `sha256sum`!

diffoscope is "just" for debugging

- Reminder: `diffoscope` is for **debugging**
- "reproducible" according to our definition means: **bit by bit identical**. So the tools for testing whether something is reproducible are either `diff` or `sha256sum`!
- `https://try.diffoscope.org`



tests.reproducible-builds.org

- Continuously testing Debian `testing`, `unstable` and `experimental`
- Also testing: coreboot, OpenWrt, LEDE, NetBSD, FreeBSD, Arch Linux, Fedora and soon F-Droid too
- 46 nodes (amd64/i386/arm64/armhf), >200 cores and >1 TB RAM
- 502 jenkins jobs running on jenkins.debian.net
- 43 scripts in Python and Bash, 283 lines of code in average
- 37 contributors for `jenkins.debian.net.git`

Variations (when testing Debian)

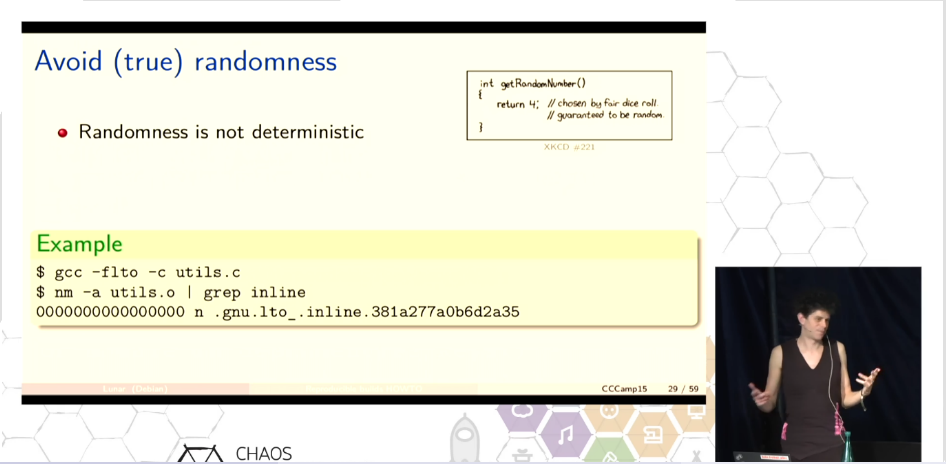
variation	first build	second build
hostname	`jenkins`	`i-capture-the-hostname`
domainname	`debian.net`	`i-capture-the-domainname`
`env TZ`	`GMT+12`	`GMT-14`
`env LANG`	`C`	`fr_CH.UTF-8`
`env LC_ALL`	not set	`fr_CH.UTF-8`
`env USER`	`pbuilder1`	`pbuilder2`
uid	`1111`	`2222`
gid	`1111`	`2222`
UTS namespace	shared with the host	*modified using /usr/bin/unshare --uts*
kernel version	Linux 3.16 or 4.X	on amd64 and arm64 always varied on armhf sometimes on i386 32/64bit kernel variation instead
umask	0022	0002
CPU type		varied on i386: Intel or AMD CPU on armhf varied a bit not varied on amd64 nor arm64
filesystem	`tmpfs`	same for both builds on amd64, i386 and arm64

Common problems

- time stamps
- timezones
- locales
- build paths
- everything else (seperated into known issues and the blurry rest)

Documentation about common problems

- `https://reproducible-builds.org/docs`
- Lunar's talk from CCCamp 2015 also on `https://media.ccc.de`



Avoid (true) randomness

- Randomness is not deterministic

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

XKCD #221

Example

```
$ gcc -flto -c utils.c
$ nm -a utils.o | grep inline
0000000000000000 n .gnu.lto_.inline.381a277a0b6d2a35
```

Lunar (Debian) Reproducible Builds HOWTO CCCamp15 29 / 59

SOURCE_DATE_EPOCH

- Build date (timestamps) usually not useful for the user
- `SOURCE_DATE_EPOCH` is defined as the last modification of the source, since the epoch (1970-01-01)
- can be used instead of current date
- can also be used for random seeds etc.
- in Debian, set from the latest `debian/changelog` entry
- can be set based on the latest git commit or the latest file modification date too

SOURCE_DATE_EPOCH

- `SOURCE_DATE_EPOCH` spec available:
- `https://reproducible-builds.org/specs/`
- many upstreams support it already
- has been adopted by other distributions (openSUSE, OpenWrt, LEDE, NetBSD, FreeBSD, Arch Linux, coreboot, Guix, ...) and many many upstreams (GCC, dpkg, rpm, mkisofs, ghostscript, libxslt, sphinx, texlive-bin, ...)

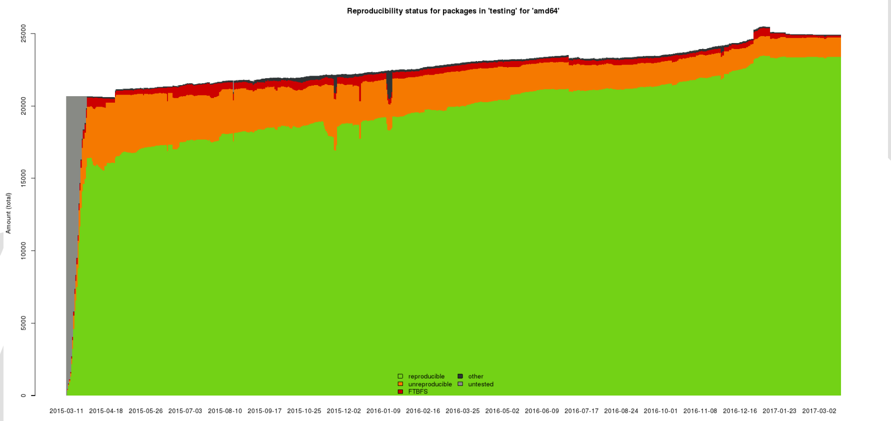
two more tools

- `strip-nondeterminism`

two more tools

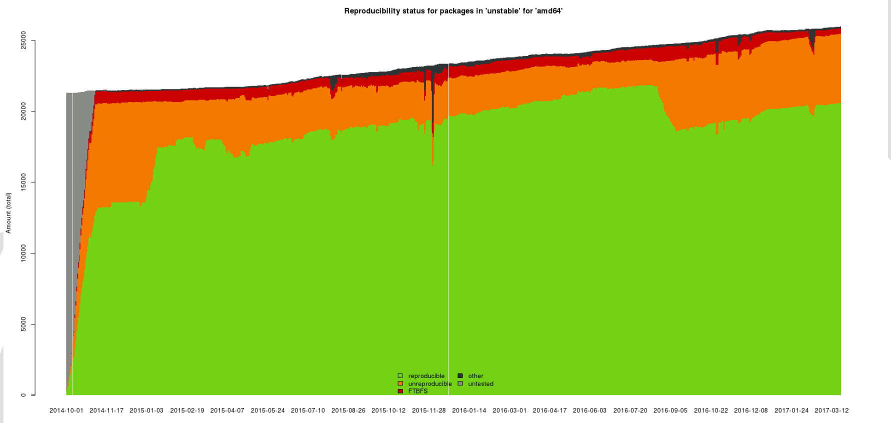
- `strip-nondeterminism`
- `reprotest`

Progress in Debian `testing` ("stretch")



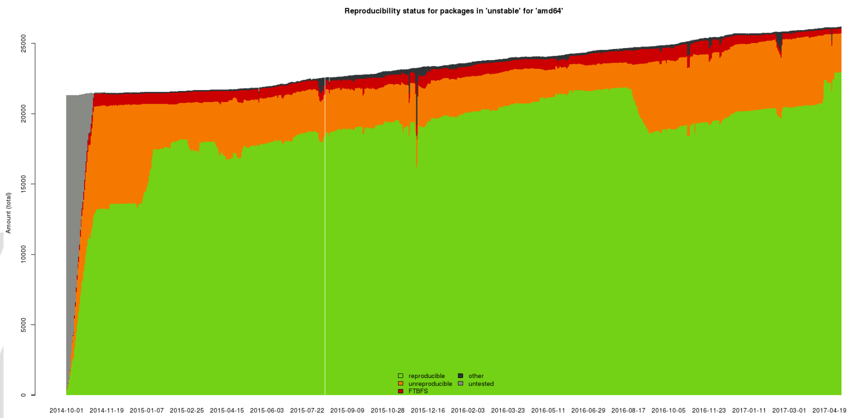
23,378 (93.8%) out of 24,909 source packages are reproducible
in our test framework on `amd64`

Progress in Debian unstable



20,597 (79.2%) out of 25,982 source packages are reproducible in our test framework on amd64 (difference due to build path variations)

Progress in Debian unstable 5 weeks later



22,950 (87.6%) out of 26,189 source packages are reproducible in our test framework on `amd64` (many build path variations resolved)

BUILD_PATH_PREFIX_MAP

- Those 93.8% in Stretch are nice, but...
- We want to be able to build in any path.
- 15-20% of the packages embed build-time paths into generated files, even though these paths do not exist at runtime, nor do they exist in the source code.
- `BUILD_PATH_PREFIX_MAP` spec available, though we have not formally released it yet...
- `https://reproducible-builds.org/specs/`
- Example patches exist, though this is still work in progress.

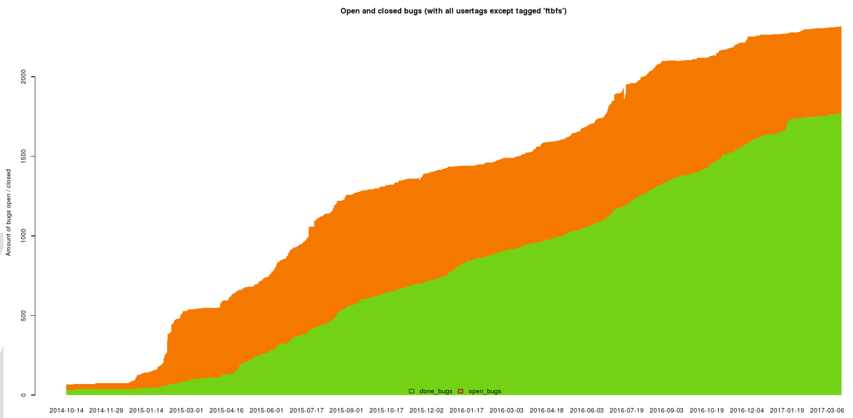
Details on tests.reproducible-builds.org

- `https://tests.reproducible-builds.org/$src`
- 48 package sets
- 292 categorised distinct issues
- 6,604 notes
- 1,473 unreproducible packages in `stretch/amd64` (testing), but only 90 without a note (5,253 in `unstable` but also only 149 without a note)
- maintained in `notes.git` by 49 contributors
- currently Debian only, but cross distro notes are planned

Debian `.buildinfo` files

- Aggregates in the same file:
  - Sources (checksums)
  - Generated binaries (checksums)
  - Packages used to build (with specific version, checksums coming soon)
- Can be later used to exactly recreate environment
- For Debian, all versions are available from `snapshot.debian.org`

Progress in the Debian bug tracker



As a rule, we file bugs with patches.
There are very few exceptions.

Sending progress upstream

- So we filed a lot of bugs... with patches...! And 1763 were even closed.
- ... but only in Debian and we rely on Debian maintainers sending them upstream.

Sending progress upstream

- So we filed a lot of bugs... with patches...! And 1763 were even closed.
- ... but only in Debian and we rely on Debian maintainers sending them upstream.
- Bernard Wiedemann (from openSUSE) thought that wasn't good enough and created `https://github.com/orgs/distropatches`

Sending progress upstream

- So we filed a lot of bugs... with patches...! And 1763 were even closed.
- ... but only in Debian and we rely on Debian maintainers sending them upstream.
- Bernard Wiedemann (from openSUSE) thought that wasn't good enough and created `https://github.com/orgs/distropatches`
- Once Debian 10, "buster" development starts, we plan to tackle those 547 open bugs too...

Debian summary / What's left to do

- This is/was a proof-of-concept, Debian is neither 93.8% reproducible nor 79.2%. (and 10% $>$ 2,500 sources packages!)

Debian summary / What's left to do

- This is/was a proof-of-concept, Debian is neither 93.8% reproducible nor 79.2%. (and 10% $>$ 2,500 sources packages!)
- All our required changes are finally in Debian now!
- Debian 9, "stretch", has 93% reproducible sources, but only one third of the binary packages are...
- Because, Debian does not (yet?) do full rebuilds before releasing... so stuff is in the archive which is not reproducible unless it's rebuild.

Debian summary / What's left to do

- This is/was a proof-of-concept, Debian is neither 93.8% reproducible nor 79.2%. (and 10% $>$ 2,500 sources packages!)
- All our required changes are finally in Debian now!
- Debian 9, "stretch", has 93% reproducible sources, but only one third of the binary packages are...
- Because, Debian does not (yet?) do full rebuilds before releasing... so stuff is in the archive which is not reproducible unless it's rebuild.
- And then we don't distribute `.buildinfo` files yet. That (and user tools) still needs more *design and code.*

Debian summary continued

- Debian 9, ”stretch”, is mostly reproducible (from source).
- Canonical can take our work now and make Ubuntu 17.10 (partially) reproducible...

Debian summary continued

- Debian 9, ”stretch”, is mostly reproducible (from source).
- Canonical can take our work now and make Ubuntu 17.10 (partially) reproducible...
- Security updates for ”stretch” can+should be reproducible!

Debian summary continued

- Debian 9, ”stretch”, is mostly reproducible (from source).
- Canonical can take our work now and make Ubuntu 17.10 (partially) reproducible...
- Security updates for ”stretch” can+should be reproducible!
- Debian 10, ”buster”, will be our first reproducible release, few exceptions expected.

Debian summary continued

- Debian 9, "stretch", is mostly reproducible (from source).
- Canonical can take our work now and make Ubuntu 17.10 (partially) reproducible...
- Security updates for "stretch" can+should be reproducible!
- Debian 10, "buster", will be our first reproducible release, few exceptions expected.
- We hope `debian-policy` will mandate 100% reproducible builds for Debian 11, "bullseye", with development starting in 2019.

Tell the world & collaborate

- ”We don’t care about Debian (only), we care about free and open source software.”

Tell the world & collaborate

- "We don't care about Debian (only), we care about free and open source software."
- 105 Weekly reports since May 2015
  - started by Lunar
  - nowadays published weekly by Chris and Ximin
  - `https://reproducible.alioth.debian.org/blog/`

Tell the world & collaborate (continued)

- First Reproducible World Summit in December 2015 (Athens, Greece)
    - `reproducible.debian.net` became `tests.reproducible-builds.org`
- Second Reproducible World Summit in December 2016 in Berlin
- Reproducible Builds Hamburg Hackathon 2017, **5-7th of May**
- Third summit in December 2017?

Tell the world & collaborate (continued)

- First Reproducible World Summit in December 2015 (Athens, Greece)
  - `reproducible.debian.net` became `tests.reproducible-builds.org`
- Second Reproducible World Summit in December 2016 in Berlin
- Reproducible Builds Hamburg Hackathon 2017, **5-7th of May**
- Third summit in December 2017?
- GSoC and Outreachy

Skipping some...

- `https://tests.r-b.org/coreboot`
- `https://tests.r-b.org/lede`
- `https://tests.r-b.org/openwrt`
- almost: `https://tests.r-b.org/f-droid`
- paused: `https://tests.r-b.org/archlinux`

Skipping some more...

- Cygnus.com (1992)
- Bitcoin (2011)
- Torbrowser (2013)

Skipping some more...

- Cygnus.com (1992)
- Bitcoin (2011)
- Torbrowser (2013)
- NixOS, GNU Guix
- ElectroBSD
- webconverger, Tails
- Google Bazil, Yocto, docker

Skipping some more...

- Cygnus.com (1992)
- Bitcoin (2011)
- Torbrowser (2013)
- NixOS, GNU Guix
- ElectroBSD
- webconverger, Tails
- Google Bazil, Yocto, docker
- ducible (build tool for Windows)
- very few commercial, propietary software
- Signal
- Shim (secure-boot)

Detour: what, reproducible commercial Software???

- Guess which

Detour: what, reproducible commercial Software???

- Guess which
- windows? (the source is available)
- medical devices in your body?
- arms?
- critical infrastructure like in nuclear powerplants?
- cars?

Detour: what, reproducible commercial Software???

- Guess which
- windows? (the source is available)
- medical devices in your body?
- arms?
- critical infrastructure like in nuclear powerplants?
- cars?
- Gambling machines!

FreeBSD vs NetBSD

- `https://tests.r-b.org/freebsd` at 99.6%
- `https://tests.r-b.org/netbsd` reached 100%

FreeBSD vs NetBSD

- `https://tests.r-b.org/freebsd` at 99.6%
- `https://tests.r-b.org/netbsd` reached 100%
- only base system built so far
- NetBSD uses non-default settings to achieve this
- ports planned

reproducible openSUSE

- `https://build.opensuse.org/package/show/home:bmwiedemann:reproducible/rpm?expand=0`
- Bernhard Wiedemann has built openSUSE twice (with some variations):
  - build-succeeded: 3172
  - bit-by-bit-identical: 2117
  - not-bit-by-bit-identical: 1055

tests.r-b.org/fedora

- used to test Fedora 23, could be made working again
- or build elsewhere and machine readable exported

Fedora basics

- `diffoscope` is available in Fedora
- `yum` and `dnf` might create non-identical environments
- `rpm`-4.13 has an option to override hostname via rpmmacros
- signed RPMs -> re-apply signature, will match for identical builds

TODO: design `.buildinfo` files from koji/mock/zypper

- rfc822 format?
- needs to define the environment
- needs to define the sources (input)
- needs to define the binaries (output)

Future work

- So far we mostly worked on making reproducible builds possible...

Future work

- So far we mostly worked on making reproducible builds possible...
- We'll need constant tests for future code.

Future work

- So far we mostly worked on making reproducible builds possible...
- We'll need constant tests for future code.
- And then, this still needs tools, infrastructure and policies to become meaningful and to be used in practice.

Rebuilds and sharing signed checksums

- Almost no work has been done here yet. We are just at the first step: being able to rebuild reproducibly...
- Different projects, different solutions?

Rebuilds and sharing signed checksums

- Almost no work has been done here yet. We are just at the first step: being able to rebuild reproducibly...
- Different projects, different solutions?
    - something like `.buildinfo` files (defining the environment, the input and the output(s)) will be needed everywhere:
    - implemented for Debian (both in sbuild and well as buildinfo.debian.net)
    - work has begun for coreboot, LEDE/OpenWrt and Fedora (mock/koji) and maybe openSUSE (OpenBuildService)

Rebuilders and sharing signed checksums, cont.

- Individuelly signed checksums (think web of trust) could work in the Debian case (we have a gpg web of trust), but IMO won't scale.
- Another idea: rebuilders, run by large organisations, eg. ACLU, BSI, CCC, Deutsche Bank, Greenpeace, GUUG, NASA, NSA, etc...
- Fedora rebuilds Debian, Debian rebuilds openSUSE, openSUSE rebuilds NetBSD, etc...
- Big customers could just rebuild everything themselves.

Integration in user tools

- "Do you really want to install this unreproducible software (y/N)"

Integration in user tools

- ”Do you really want to install this unreproducible software (y/N)”
- ”Do you want to build those packages which have unconfirmed checksums, before installing? (Y/n)”

Integration in user tools

- ”Do you really want to install this unreproducible software (y/N)”
- ”Do you want to build those packages which have unconfirmed checksums, before installing? (Y/n)”
- ”How many signed checksums do you require to call a package ’reproducible’?” - and whom do you trust?

As a software developer

- Stop using build dates

As a software developer

- Stop using build dates
- Stop using build dates

As a software developer

- Stop using build dates
- Stop using build dates
- Do you really need build dates?

As a software developer

- Stop using build dates
- Stop using build dates
- Do you really need build dates?
- Seriously?

As a software developer

- Stop using build dates
- Stop using build dates
- Do you really need build dates?
- Seriously?
- Then use `SOURCE_DATE_EPOCH` instead
- See `https://reproducible-builds.org/specs/source-date-epoch/`

Form your reproducible builds team!

- Why?
  - Every distribution should be reproducible!
  - Learn something new everyday
  - Change the (software) world!
  - `https://tests.reproducible-builds.org/$distro` needs **your** help
- How to get started?
  - Build something twice, run `diffoscope` on the results.
  - Experiment - learning by doing
  - RTFM, there is lots of documentation
  - Talk to me here or talk to us on IRC or via mail.

Thank You!

- All "Reproducible Builds" contributors
  (You are just **so** awesome!)
- **Dario Cavedon** for doing the by far greatest part of organization for the DUCC-IT

```
mattia@debian.org  66AE 2B4A FCCF 3F52 DA18
                   4D18 4B04 3FCD B944 4540
```

Questions, comments, ideas?

- `https://reproducible-builds.org/`
- `#reproducible-builds` on `irc.OFTC.net`
- `https://lists.reproducible-builds.org`
- twitter: @ReproBuild