Reproducible Builds

Valerie Young
(spectranaut)

Linux Conf Australia 2016

Reproducible Builds

What if you could always compile free software?

Valerie Young
(spectranaut)

Linux Conf Australia 2016

Valerie Young

- F96E 6B8E FF5D 372F FDD1 DA43 E8F2 1DB3 3D9C 12A9
- spectranaut on OFTC/freenode
- Studied physics and computer science at BU (2012)
- Programmer at athenahealth
- Ubuntu/Debian user since 2012
- Debian contributor since May 2016

...Thanks to Outreachy!

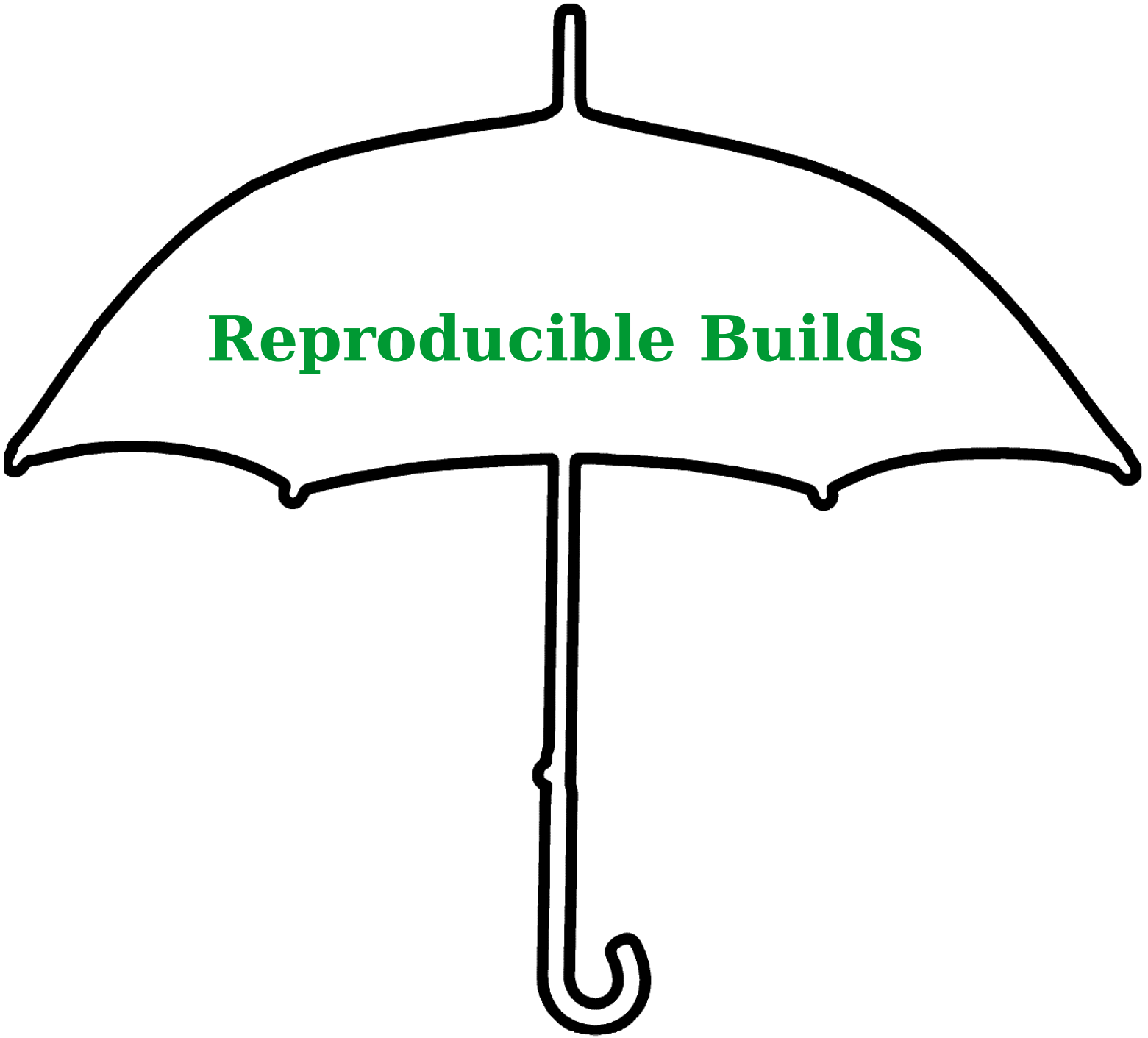
- **outreachy.gnome.org**
- Funding for women and minorities to work on free software
- 3 month projects (like Google summer of code)
- 3 month (and beyond) free software mentor
- Not limited to programming

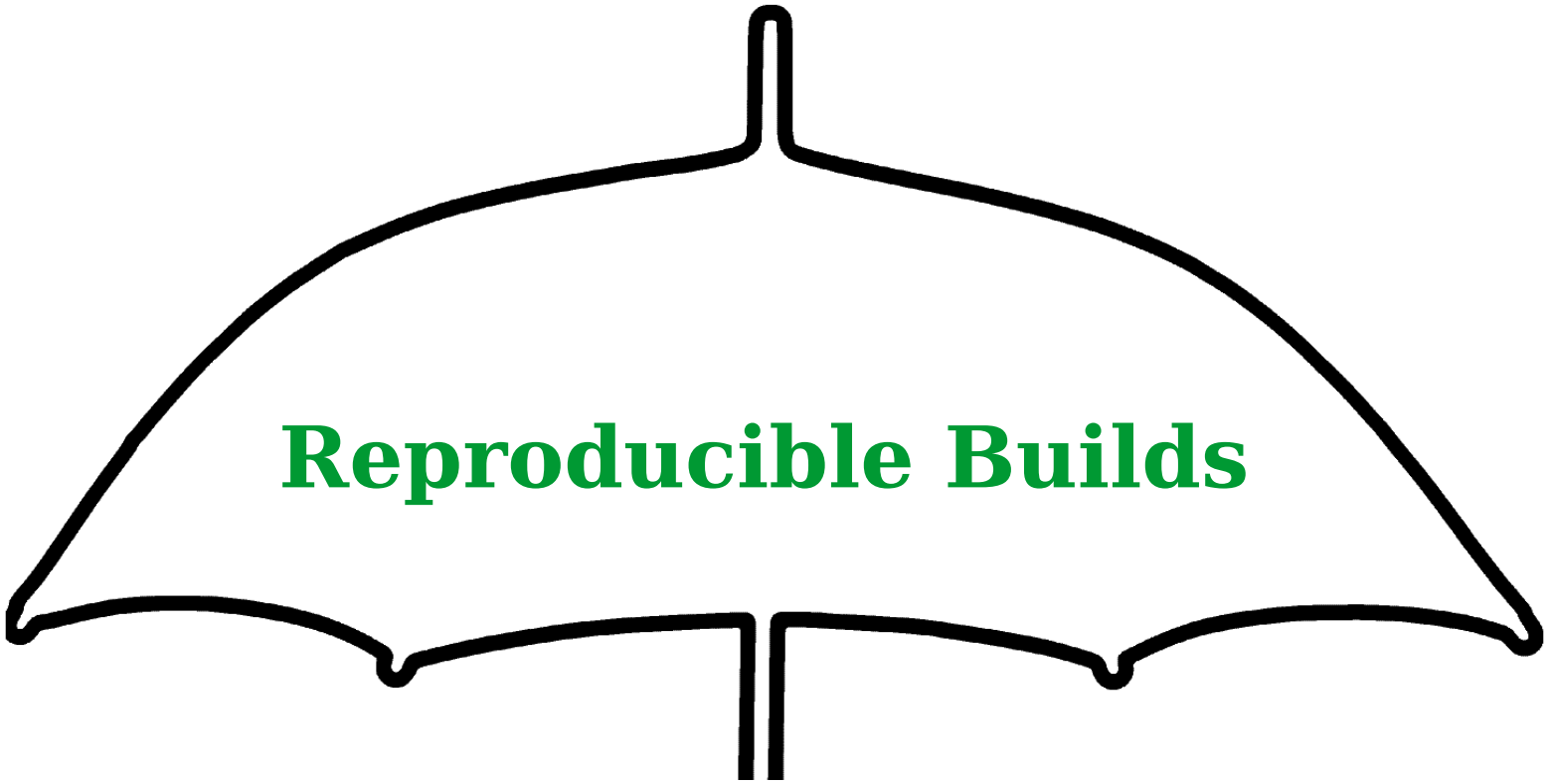
Overview

1. What is “Reproducible Builds”?
2. Reproducible builds effect on software freedoms
3. Up-to-date history of reproducible builds efforts
4. What is left to do..?

Overview

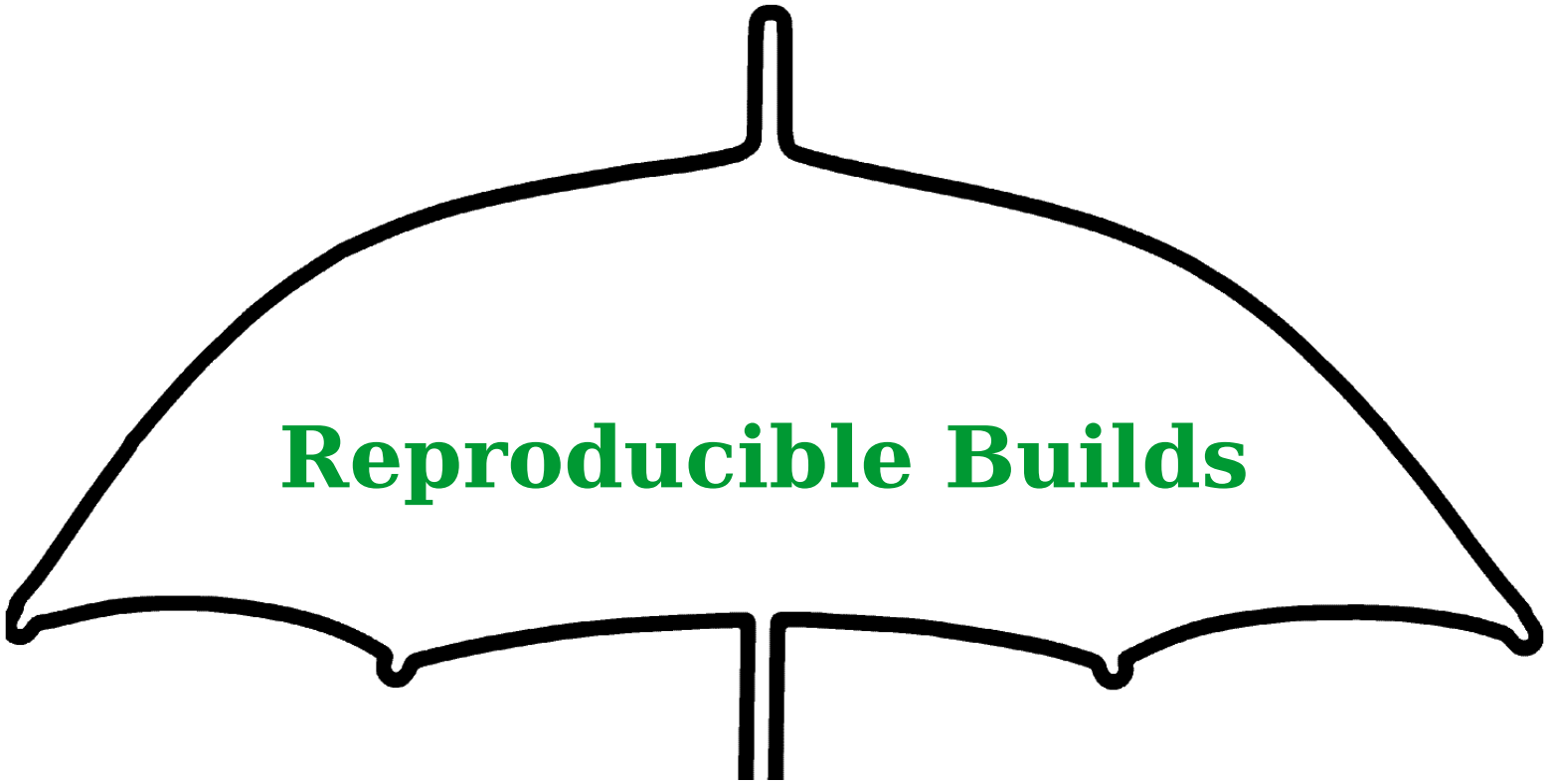
1. **What is “Reproducible Builds”?**
2. Reproducible builds effect on software freedoms
3. Up-to-date history of reproducible builds efforts
4. What is left to do..?


Reproducible Builds



Goals:

1. Compilation of binary should be deterministic



Goals:

1. Compilation of binary should be deterministic

2. Build environment of binary should be reproducible

Overview

1. What is “Reproducible Builds”?
2. **Reproducible builds effect on software freedoms**
3. Up-to-date history of reproducible builds efforts
4. What is left to do..?

Software Freedoms

- (0) The freedom to **run** the program for any purpose.
- (1) The freedom to **study** how the program works, and **change** it to your needs.
- (2) The freedom to **redistribute** copies so you can help your neighbor.
- (3) The freedom to improve the program, and **release your improvements** to the public, so that the whole community benefits.

Software Freedoms

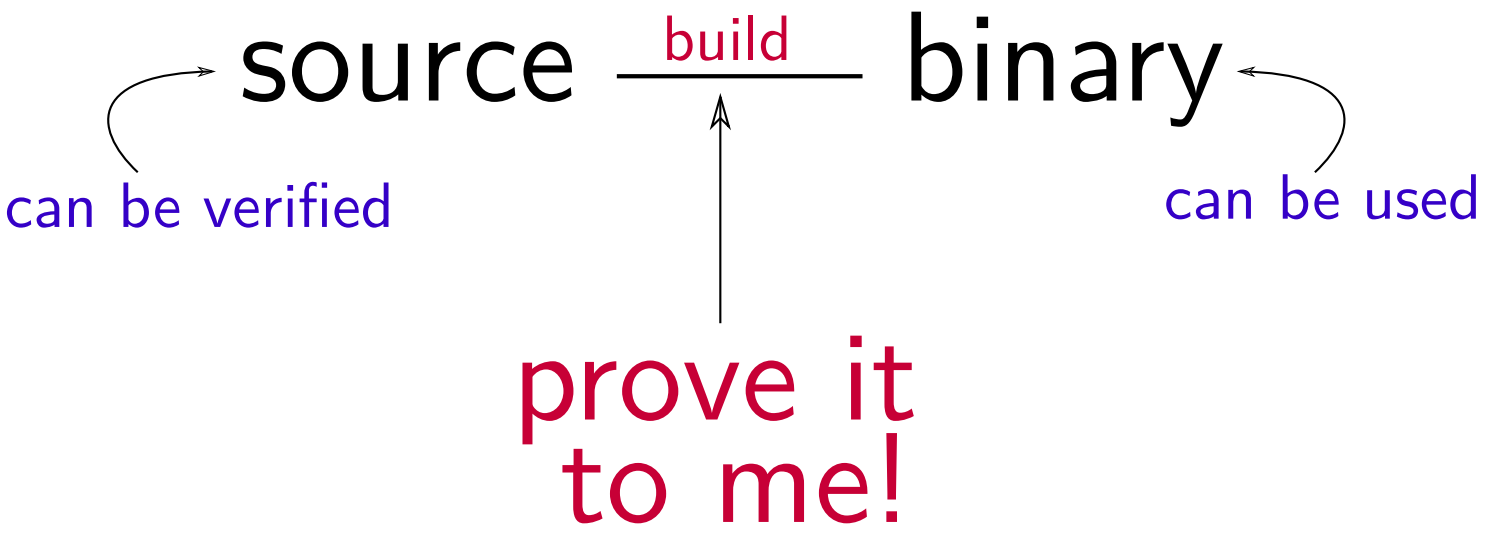
- (0) The freedom to **run** the program for any purpose.
- (1) The freedom to **study** how the program works, and **change** it to your needs.
- (2) The freedom to **redistribute** copies so you can help your neighbor.
- (3) The freedom to improve the program, and **release your improvements** to the public, so that the whole community benefits.

Freedom 1a:
Can we **study** the program?

Freedom 1a:
Can we **study** the program?

source —build→ binary

Freedom 1a:
Can we **study** the program?



Freedom 1a:
Can we **study** the program?

- Not without faith.. or bit-for-bit reproducibility!

Freedom 1a:
Can we **study** the program?

- Not without faith.. or bit-for-bit reproducibility!
- Even one bit can compromise a computer
  - OpenSSH (CVE-2002-0083)

Freedom 1a:
Can we **study** the program?

- Not without faith.. or bit-for-bit reproducibility!
- Even one bit can compromise a computer
  - OpenSSH
- Without reproducible builds, the developer is single point of failure
  - Compromised human or machines

For more security motivation, see:
https://events.ccc.de/congress/2014/Fahrplan/events/6240.html

Freedom 1b:
Can we **change** the program?

source —build→ binary

Freedom 1b:
Can we **change** the program?

- Not without great difficulty… or reproducible builds!

Freedom 1b:
Can we **change** the program?

- Not without great difficulty… or reproducible builds!
- “Build environment should be reproducible”
  - Lower barrier to contribution for lazy people

Freedom 1b: Can we **change** the program?

- Not without great difficulty… or reproducible builds!
- “Build environment should be reproducible”
  - Lower barrier to contribution for lazy people
- Arguably, code is easier to edit than compile
  - Lower barrier to contribution for non-technical, competent people (designers? User researchers?)

Overview

1. What is “Reproducible Builds”?
2. Reproducible builds effect on software freedoms
3. **Up-to-date history of reproducible builds**
4. What is left to do..?

How to change 60 years of non-deterministic programming habits?

- Since 2012
- Why?
  - $$$
- Created Gitian
  - Build in VM
- Removes indeterminacies:
  - Compiler versions
  - Kernel versions
  - Build machine meta-data (hostname, time)

- Reproducibly built since 2012
- Why?
  - Human lives.
- More complex
  - Firefox browser
  - And 50+ packages
- Used Gitian
  - And a few months of developing..

What else did Tor find?

- Python os.walk: Multi-threaded build processes results in random file ordering.
- GNU binutils: Consistently random bits... that result from uninitialized memory.

More fun Tor reproducibility facts:
https://blog.torproject.org/blog/deterministic-builds-part-two-technical-details

What else did Tor find?

- Python os.walk: Multi-threaded build processes results in random file ordering.
- GNU binutils: Consistently random bits... that result from uninitialized memory.

Problems they could not solve:

- Takes a long time
- Browser profile-guided optimizations

More fun Tor reproducibility facts:
https://blog.torproject.org/blog/deterministic-builds-part-two-technical-details

Think reproducing Tor sounds hard?

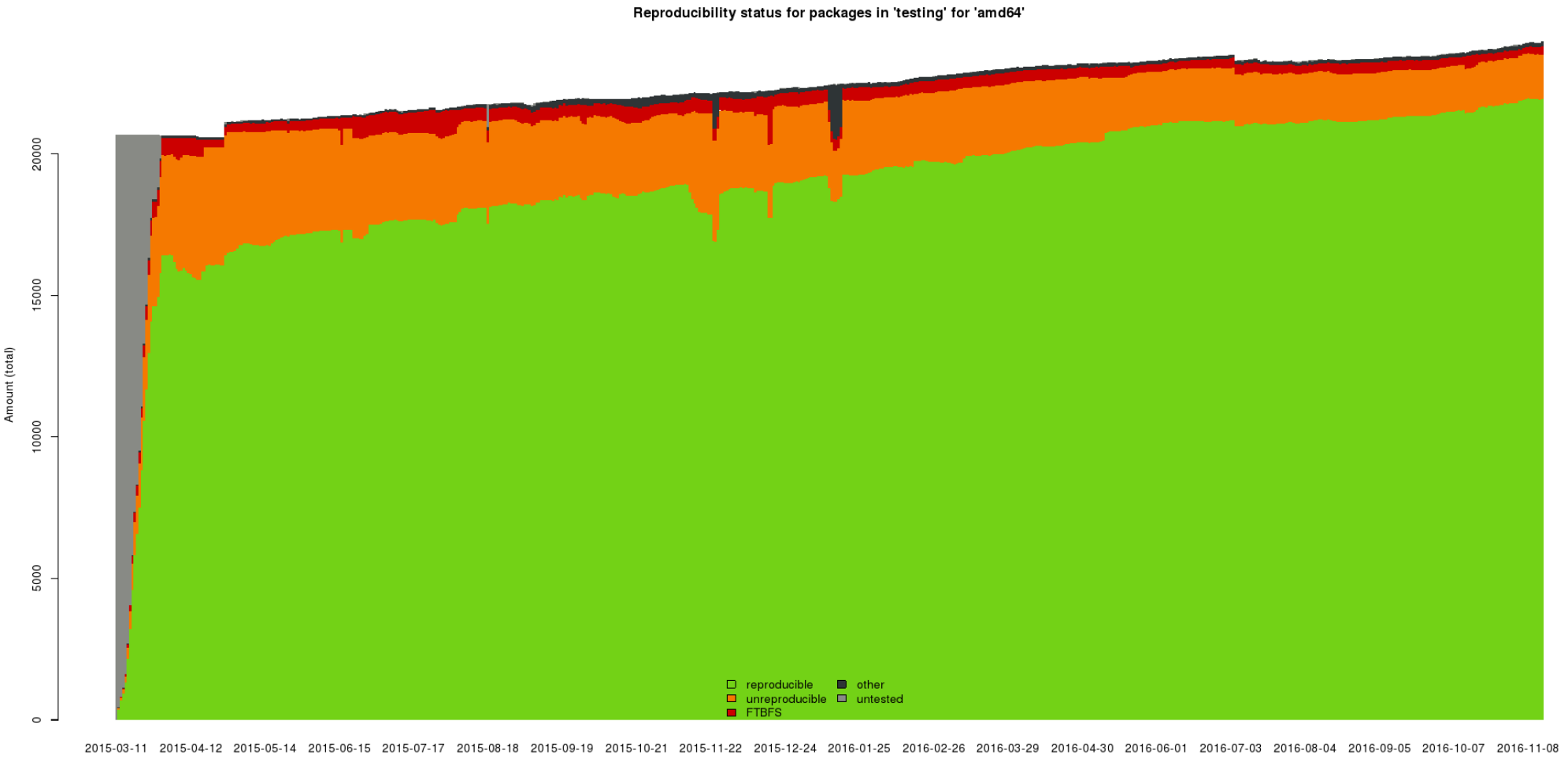
debian

- >40,000 packages
- ~1000 developers
- All the languages..
- ..all the compilers.

How to began:

- A discussion at DebConf13 and a wikipage
- Attempts to prove reproducibility of a few packages
- Quickly realized maybe problems existed in packaging toolchain
- End of 2014 saw the beginning of continuous testing of all packages

tests.reproducible-builds.org



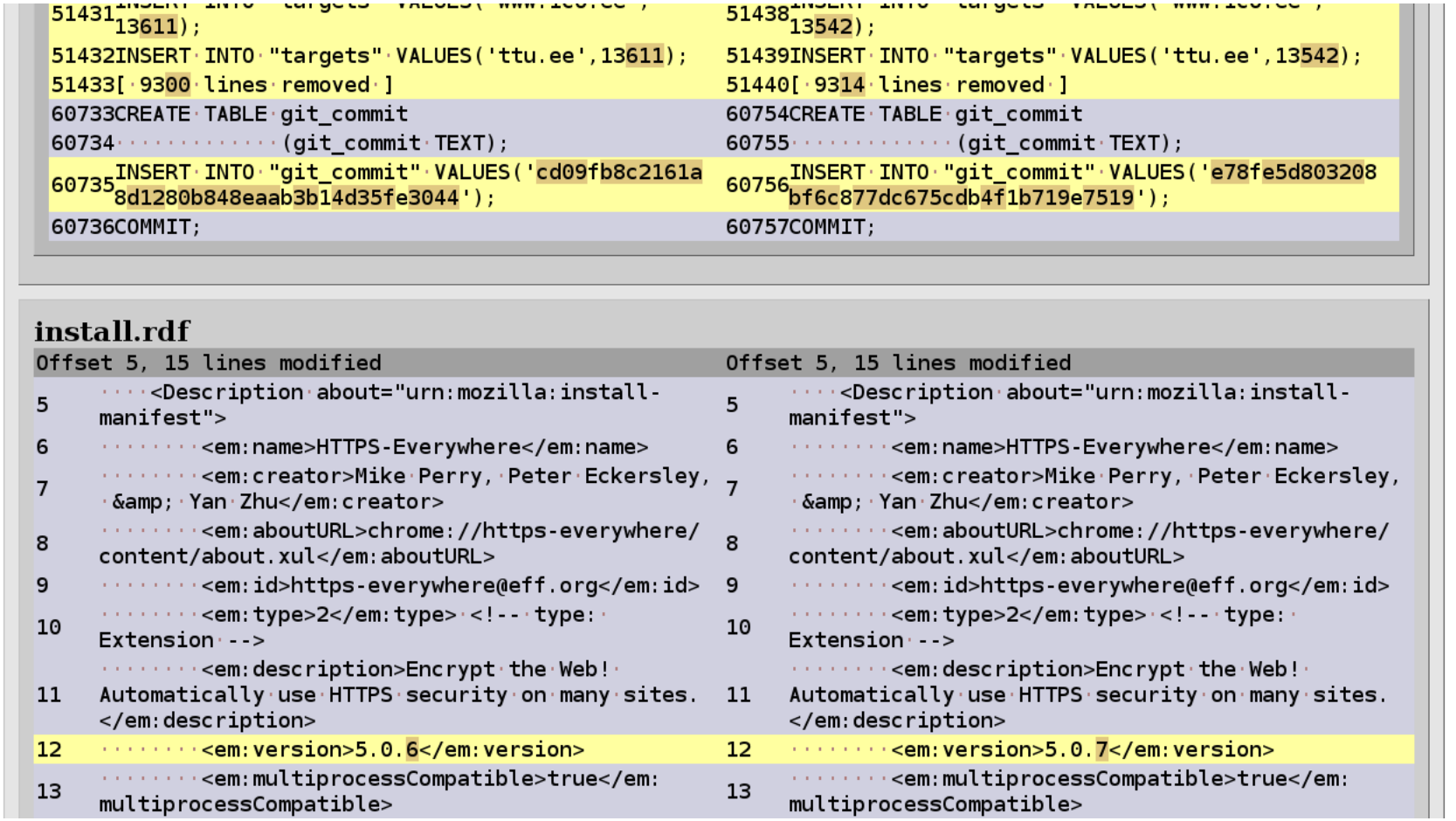
tests.reproducible-builds.org/<**package**>

 Reproducible  Unreproducible

- Test = building twice and comparing
- Testing on amd64, arm and i386
- Variations between builds:
  - domain
  - hostname
  - timezone
  - language
  - locale
  - time
  - user
  - program id
  - shell
  - kernel
  - cpu type
  - file ordering

Unreproducible Packages

Diffoscope

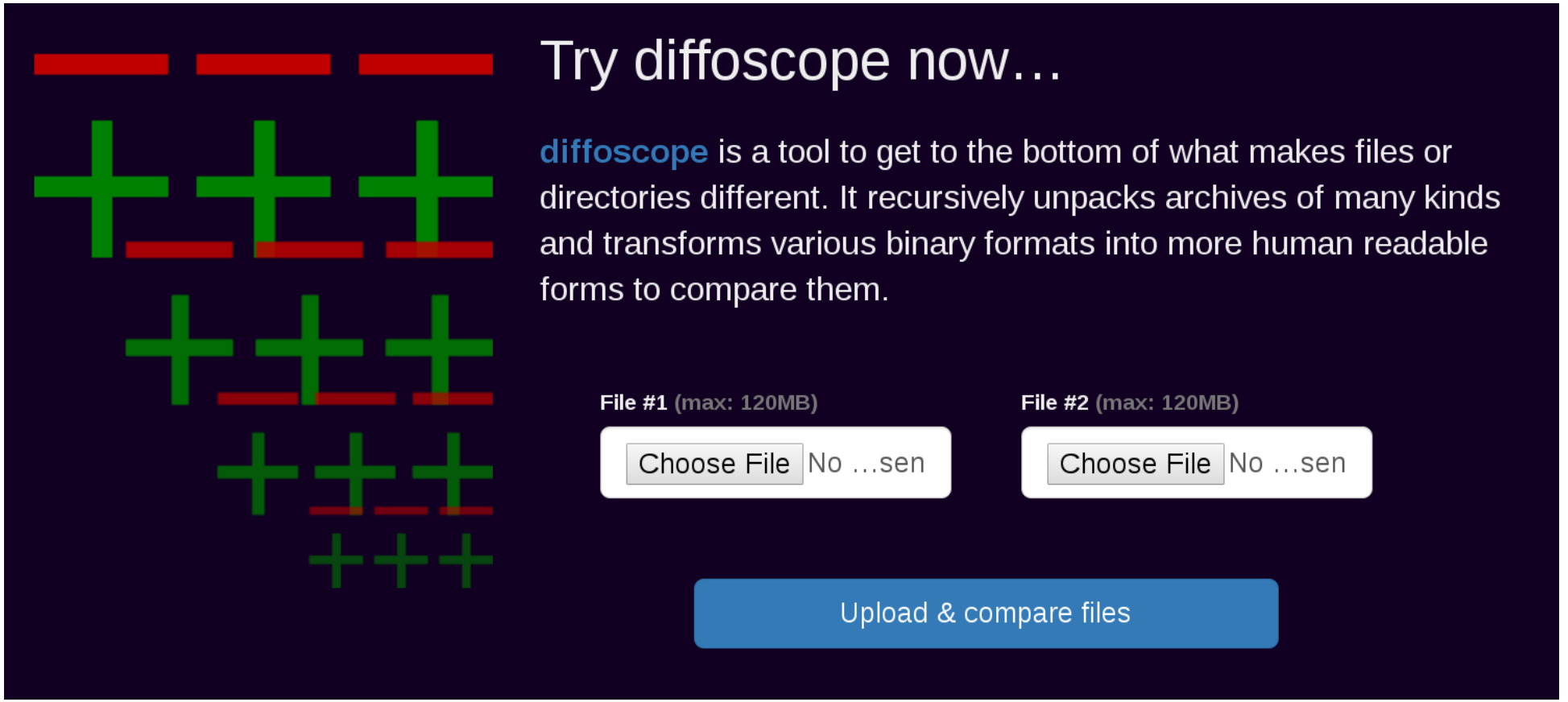


51431	13611);	51438	13542);
51432	INSERT INTO "targets" VALUES('ttu.ee',13611);	51439	INSERT INTO "targets" VALUES('ttu.ee',13542);
51433	[9300 lines removed]	51440	[9314 lines removed]
60733	CREATE TABLE git_commit	60754	CREATE TABLE git_commit
60734	(git_commit TEXT);	60755	(git_commit TEXT);
60735	INSERT INTO "git_commit" VALUES('cd09fb8c2161a8d1280b848eaab3b14d35fe3044');	60756	INSERT INTO "git_commit" VALUES('e78fe5d803208bf6c877dc675cdb4f1b719e7519');
60736	COMMIT;	60757	COMMIT;

install.rdf

Offset 5, 15 lines modified		Offset 5, 15 lines modified	
5	<Description about="urn:mozilla:install-manifest">	5	<Description about="urn:mozilla:install-manifest">
6	<em:name>HTTPS-Everywhere</em:name>	6	<em:name>HTTPS-Everywhere</em:name>
7	<em:creator>Mike Perry, Peter Eckersley, & Yan Zhu</em:creator>	7	<em:creator>Mike Perry, Peter Eckersley, & Yan Zhu</em:creator>
8	<em:aboutURL>chrome://https-everywhere/content/about.xul</em:aboutURL>	8	<em:aboutURL>chrome://https-everywhere/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: Extension -->	10	<em:type>2</em:type> <!-- type: Extension -->
11	<em:description>Encrypt the Web! Automatically use HTTPS security on many sites.</em:description>	11	<em:description>Encrypt the Web! Automatically use HTTPS security on many sites.</em:description>
12	<em:version>5.0.6</em:version>	12	<em:version>5.0.7</em:version>
13	<em:multiprocessCompatible>true</em:multiprocessCompatible>	13	<em:multiprocessCompatible>true</em:multiprocessCompatible>

https://try.diffoscope.org



Unreproducible Packages

Issue Tracking

- We have “notes” for most unreproducible packages
- 261 distinct issues tagged in notes.git
  - Described in issues.git
  - Examples: `timestamps_in_zip`, `captures_build_path`, `different_encoding`
- Many incredible Debian developers and contributors up keep these notes.
  - Filed >2000 bugs with patches
  - Filed >3000 bugs that fail to build with new libs

TIMESTAMPS

- 112 issues are related to recording the time of the build in the binary.
  - Need build timestamps for documentation?
  - Need build timestamps for reconstructing build env?
  - Need builds timestamps for randomness seed?
  - Need build times stamps for ...?

TIMESTAMPS

- 112 issues are related to recording the time of the build in the binary.
  - Need build timestamps for documentation?
  - Need build timestamps for reconstructing build env?
  - Need builds timestamps for randomness seed?
  - Need build times stamps for ...?

Nope, you don't!

TIMESTAMPS

- Debian recommends: **SOURCE_DATE_EPOCH**
  - Set to the last time the source was changed
  - Specification has been written for upstream developers
  - Many have followed:
    - Debhelper, epydoc, ghostscript, ocamldoc…
    - In discussion: GCC for `__DATE__` and `__TIME__` macros

Additional projects

- Testing: OpenWRT, coreboot, NetBSD, FreeBSD
- Almost testing: ArchLinux, Fedora and F-Driod

More information

- reproducible-builds.org
- Lunar talk on “How to make your software reproducible” at Chaos Communication Camp 2015

Overview

1. What is “Reproducible Builds”?
2. Reproducible builds effect on software freedoms
3. Recent history of reproducible builds
4. **What is left to do..?**

“Reproduced Builds” are not enough

Part I

- Debian is 0% reproducible until *any user* can reproduce any given binary Debian package.
- “Build environment should be reproducible”

Build environment metadata: Debian's **.buildinfo** files

- .buildinfo files contain:
  - Checksum of the source
  - Checksum of generated binaries
  - Exact versions of all build dependencies
- Left to do: distribute .buildinfo files

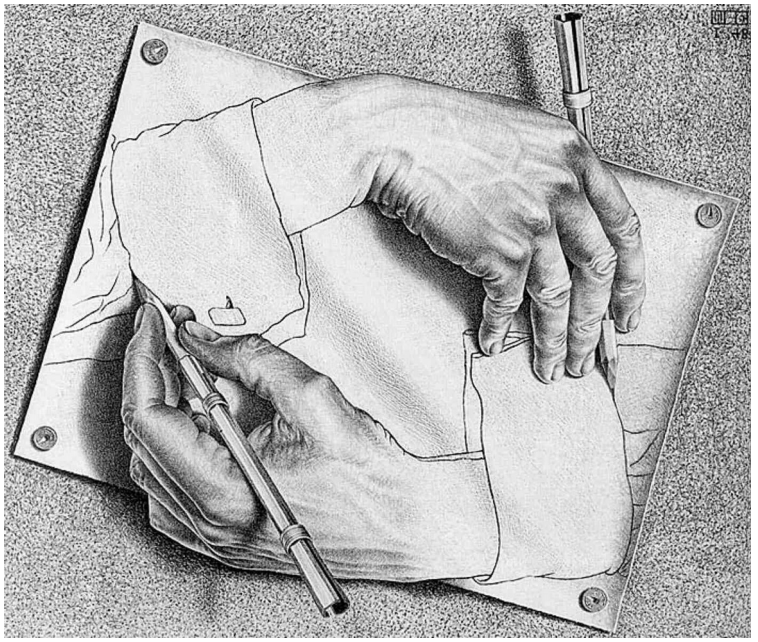
.buildinfo file

```
Format: 1.9
Build-Architecture: amd64
Source: txtorcon
Binary: python-txtorcon
Architecture: all
Version: 0.11.0-1
Build-Path: /build/txtorcon-
0.11.0-1
Checksums-Sha256:
 a26549d9…7b 125910 python-
txtorcon_0.11.0-1_all.deb
 28f6bcbe…69 2039 txtorcon_0.11.0-
1.dsc
Build-Environment:
 base-files (= 8),
 base-passwd (= 3.5.37),
 bash (= 4.3-11+b1),
 …
```

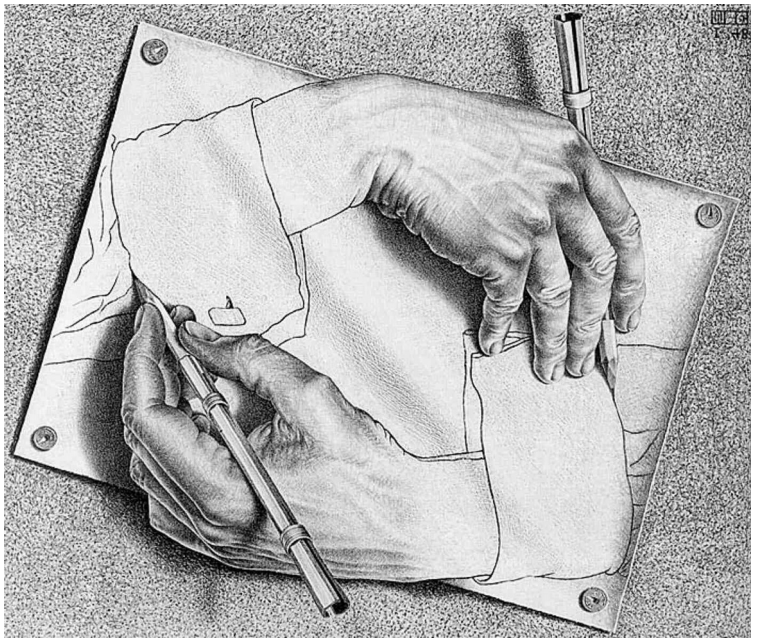
Build environment metadata: Can you verify the builds?

- We need tools to re-create build environment
  - Debian: can use .buildinfo files and archive.debian.net
  - other distros: ...?

Delivering build environment metadata with software..



Delivering build environment metadata with software..



Delivers the **freedom** to **modify** software.

With this software freedom, what else do we get?

- Guaranteed compilation → more contributors!

With this software freedom, what else do we get?

- Guaranteed compilation → more contributors!
- Easier regulation..
  - Allows audits of binaries
  - Presently unaudited binaries include: voting software, VW emission scandal...
- Easier GPL enforcement

With this software freedom, what else do we get?

- Guaranteed compilation → more contributors!
- Easier regulation..
  - Allows audits of binaries
  - Presently unaudited binaries include: voting software, VW emission scandal...
- Easier GPL enforcement
- Perhaps a more long term preference for free software?

“Reproduced Builds” are not enough

Part II

- How can we surface verified reproducibility to a non-developer?

Debian: Uploading and Verifying

- Who will rebuild software?
  - Dedicated rebuilders
  - Other developers
- Sign and share the signatures on binaries
  - “web of trust” solution probably won't scale

Debian:
Downloading and Verifying

```
Do you really want to install this
unreproducible software? (y/N)
```

Debian:
Downloading and Verifying

```
Do you really want to install this
unreproducible software? (y/N)
```

```
Do you want to build these packages with
unconfirmed checksums before installing? (Y/n)
```

Debian: Downloading and Verifying

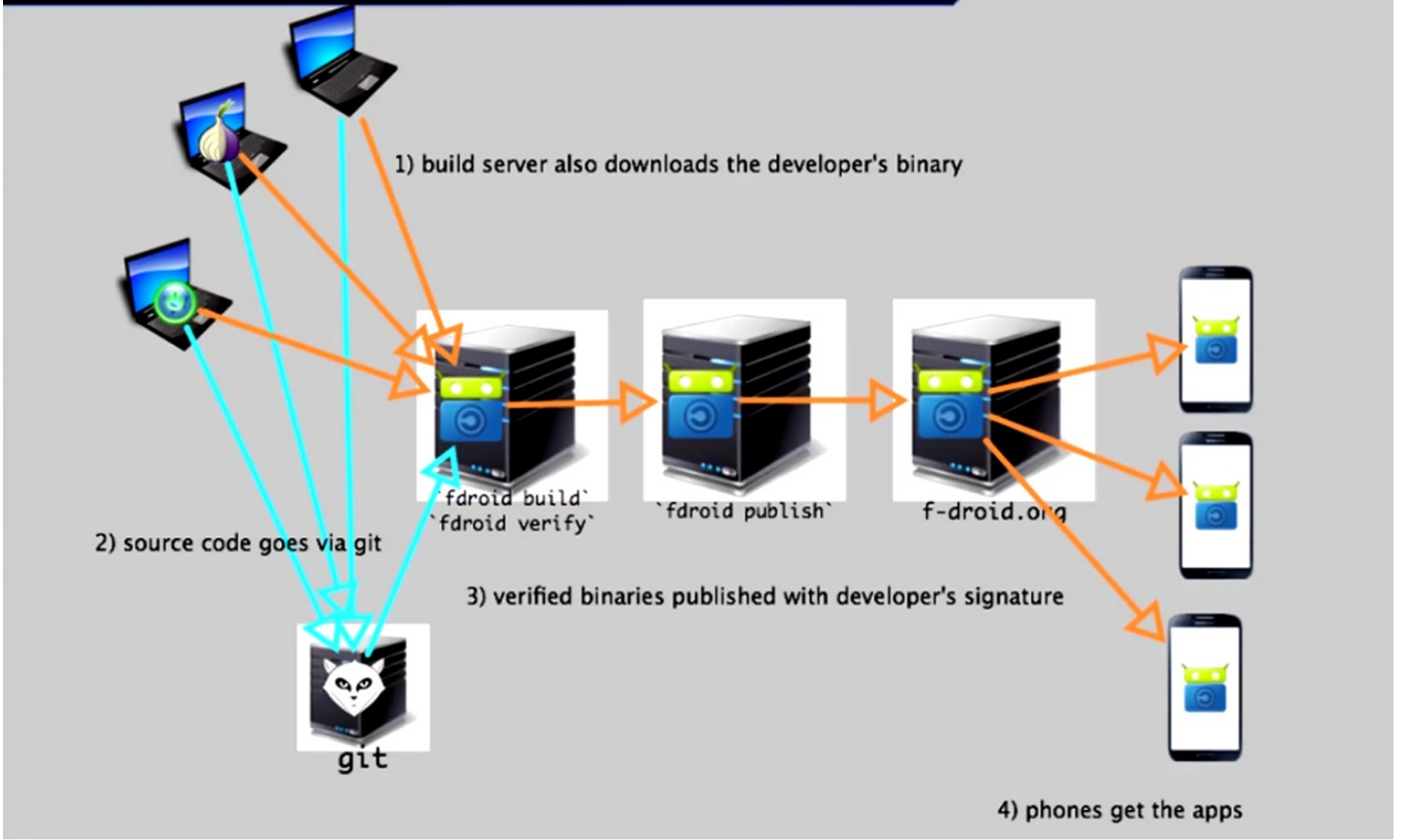
```
Do you really want to install this
unreproducible software? (y/N)
```

```
Do you want to build these packages with
unconfirmed checksums before installing? (Y/n)
```

```
How many signed checksums do you require to call
a package “reproducible”?
```

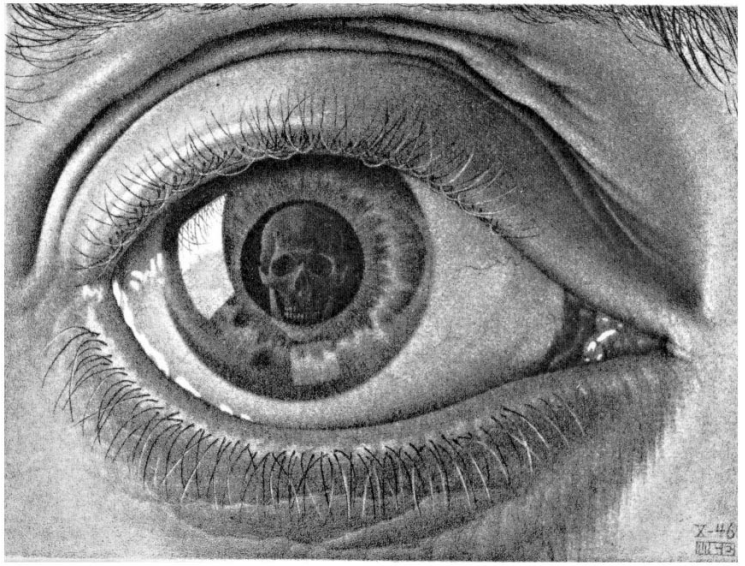
```
Which rebuilders do you trust?
```

FDroid reproducible process

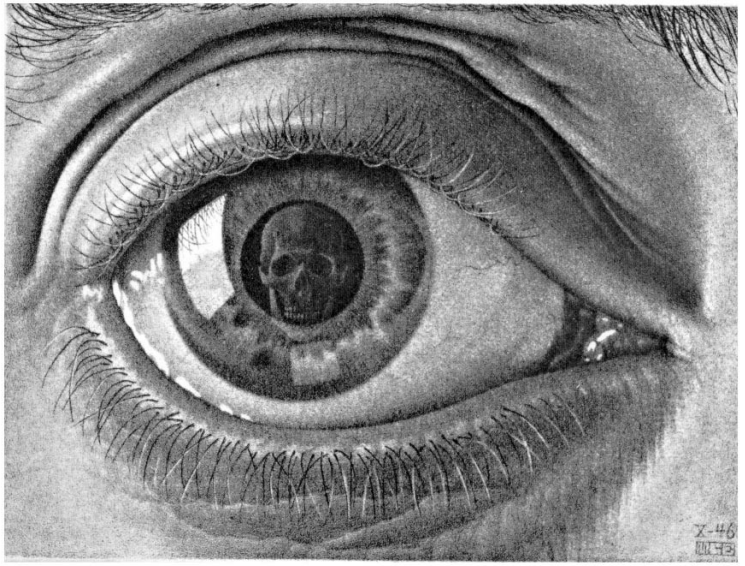


https://events.ccc.de/congress/2014/Fahrplan/events/6240.html

Delivering the verification of reproducibility with binaries..



Delivering the verification of reproducibility with binaries..



Delivers the **trust** we have in free software because we can **study** the source.

With this software freedom, what do we get?

- Assurance against compromised developers
- Assurance against compromised compilers
  - Unless you compromise them all!
- Free software = provably safer and more transparent than proprietary.

Thanks!

More information:
reproducible-builds.org
#reproducible-builds on OFTC