# Reproducible Builds in FreeBSD

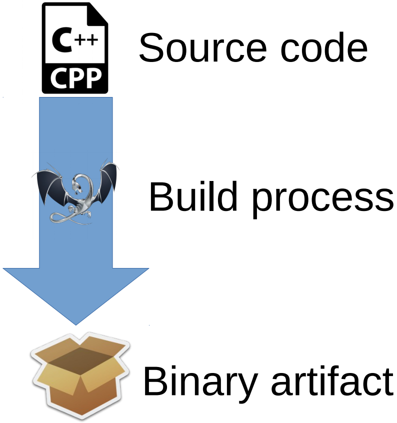
Ed Maste
emaste@freebsd.org

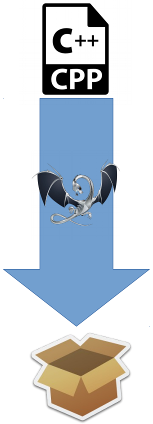
AsiaBSDCon 2017
Tokyo, Japan
2017-03-12

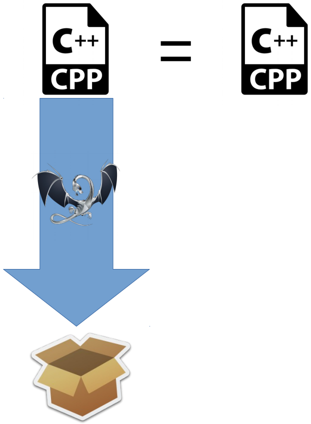
## About Me

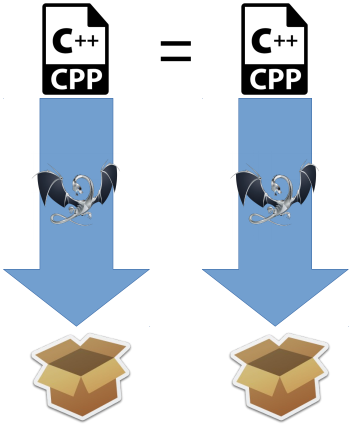
- FreeBSD user since early 2000s
- FreeBSD committer since 2005
- FreeBSD Foundation since 2011
- Reproducible builds since 2015

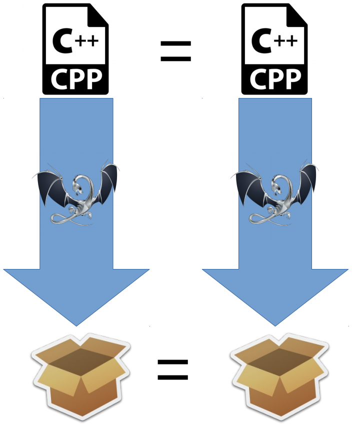
Source code


CPP
Source code
Build process
Binary artifact


C++
CPP


C++
CPP
=
C++
CPP


C++
CPP
=
C++
CPP


C++
CPP
=
C++
CPP
=

## Definition

A build is **reproducible** if given the **same source code**, **build environment**, and **build instructions**, **any party** can recreate **bit-by-bit identical** copies of all specified artifacts.

The relevant attributes of the build environment, the build instructions, and the source code as well as the expected reproducible artifacts are defined by the authors or distributors. The artifacts of a build are the parts of the build results that are the desired, primary output.

https://reproducible-builds.org/docs/definition/

## Why?

- Software Integrity / Assurance
- Practical Reasons

## Practical Reasons

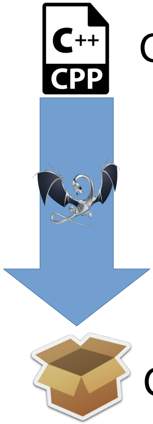
- Reduce package repository storage size (if storing every build)
- Reduce bandwidth used by package mirrors and end host updates
- Create minimal-size binary patches
- Facilitate retroactive debug data creation
- Accurate ports tree exp-runs – track:
  - Changing toolchain components
  - Packages impacted by a change in a headers or macros
  - Packages using static libraries
  - Improved packaging Q/A

## Software Assurance

“**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.”

https://reproducible-builds.org/

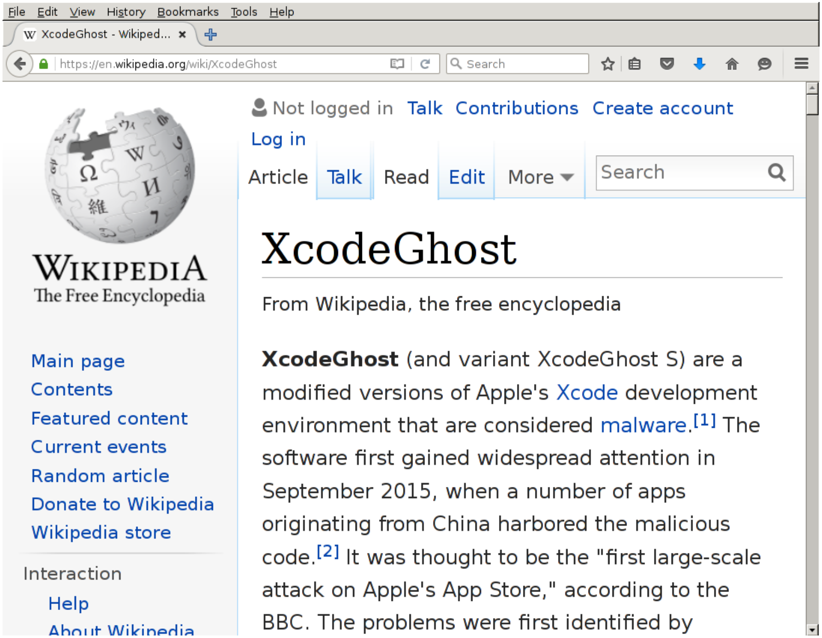
## Software Assurance



Can be audited

Can be used

# XCode Malware

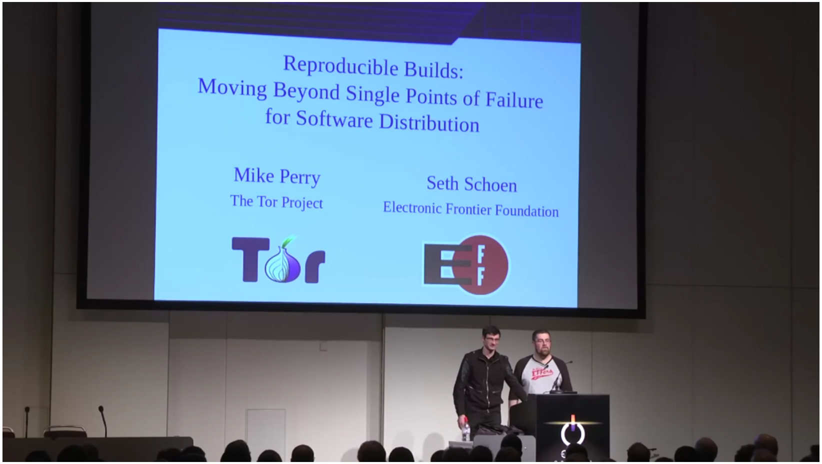

File Edit View History Bookmarks Tools Help
XcodeGhost - Wikiped...
https://en.wikipedia.org/wiki/XcodeGhost
Search
Not logged in Talk Contributions Create account Log in
Article Talk Read Edit More
Search
Wikipedia The Free Encyclopedia
Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store
Interaction
Help
About Wikipedia

XcodeGhost

From Wikipedia, the free encyclopedia

**XcodeGhost** (and variant XcodeGhost S) are a modified versions of Apple's Xcode development environment that are considered malware.[1] The software first gained widespread attention in September 2015, when a number of apps originating from China harbored the malicious code.[2] It was thought to be the "first large-scale attack on Apple's App Store," according to the BBC. The problems were first identified by

## 31c3 Perry and Schoen



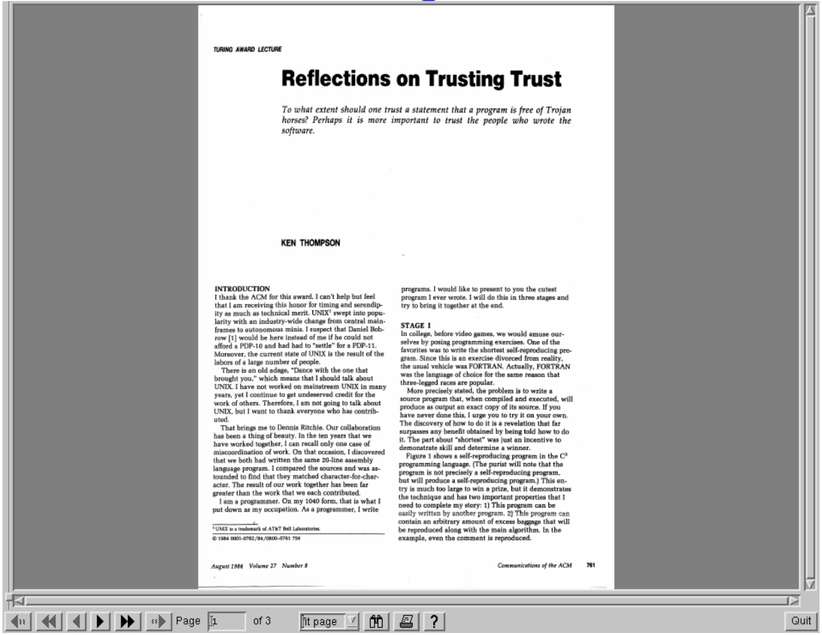
Reproducible Builds:
Moving Beyond Single Points of Failure
for Software Distribution

Mike Perry
The Tor Project

Seth Schoen
Electronic Frontier Foundation

https://media.cc.de "Moving Beyond Single Points of Failure"

# Reflections on Trusting Trust



TURING AWARD LECTURE

## Reflections on Trusting Trust

*To what extent should one trust a statement that a program is free of Trojan horses? Perhaps it is more important to trust the people who wrote the software.*

KEN THOMPSON

### INTRODUCTION

I thank the ACM for this award. I can't help but feel that I am receiving this honor for timing and serendipity as much as technical merit. UNIX[1] swept into popularity with an industry-wide change from central mainframes to autonomous minis. I suspect that Daniel Bobrow [1] would be here instead of me if he could not afford a PDP-10 and had had to "settle" for a PDP-11. Moreover, the current state of UNIX is the result of the labors of a large number of people.

There is an old adage, "Dance with the one that brought you," which means that I should talk about UNIX. I have not worked on mainstream UNIX in many years, yet I continue to get undeserved credit for the work of others. Therefore, I am not going to talk about UNIX, but I want to thank everyone who has contributed.

That brings me to Dennis Ritchie. Our collaboration has been a thing of beauty. In the ten years that we have worked together, I can recall only one case of miscoordination of work. On that occasion, I discovered that we both had written the same 20-line assembly language program. I compared the sources and was astounded to find that they matched character-for-character. The result of our work together has been far greater than the work that we each contributed.

I am a programmer. On my 1040 form, that is what I put down as my occupation. As a programmer, I write programs. I would like to present to you the cutest program I ever wrote. I will do this in three stages and try to bring it together at the end.

### STAGE I

In college, before video games, we would amuse ourselves by posing programming exercises. One of the favorites was to write the shortest self-reproducing program. Since this is an exercise divorced from reality, the usual vehicle was FORTRAN. Actually, FORTRAN was the language of choice for the same reason that three-legged races are popular.

More precisely stated, the problem is to write a source program that, when compiled and executed, will produce as output an exact copy of its source. If you have never done this, I urge you to try it on your own. The discovery of how to do it is a revelation that far surpasses any benefit obtained by being told how to do it. The part about "shortest" was just an incentive to demonstrate skill and determine a winner.

Figure 1 shows a self-reproducing program in the C[3] programming language. (The purist will note that the program is not precisely a self-reproducing program, but will produce a self-reproducing program.) This entry is much too large to win a prize, but it demonstrates the technique and has two important properties that I need to complete my story: 1) This program can be easily written by another program. 2) This program can contain an arbitrary amount of excess baggage that will be reproduced along with the main algorithm. In the example, even the comment is reproduced.

[1] UNIX is a trademark of AT&T Bell Laboratories.

© 1984 0001-0782/84/0800-0761 75¢

## Who is Involved

## Who's Involved

- F-Droid
- Bitcoin
- Tor
- Signal
- OpenSUSE
- Ubuntu
- Guix
- NixOS
- ElectroBSD
- Qubes
- TAILS
- Subgraph

## Components of Reproducible Builds

- Deterministic build system
- Reproducible build environment
- Distributing the build environment
- Rebuilding and checking the results

## Deterministic build system

- Stable inputs
- Stable outputs
- Capture as little as possible from the environment

## Reproducing the build environment

- Build tools and versions
- Build architecture
- Operating system
- Build path
- Build date and time
- ...
- *FreeBSD base system provides a shortcut*

## Distributing the build environment

- Fetch and build known toolchain
- Integrated toolchain source
- Packaged toolchain
- Gitian
- Containers
- VM images
- ...
- *FreeBSD base system provides a shortcut*

# Sources of nonreproducibility

- Embedded build information
- Input file ordering (filesystem, locale)
- Archive metadata
- Unstable output ordering (e.g. hashes)
- Intentional randomness
- DWARF debug info paths
- Threaded producers
- Optimizations
- Value initialization
- Embedded signatures

## Variations

- hostname, domainname
- Environment: `TZ`, `LANG`, `LC_ALL`, `USER`
- Timestamp, Y/M/D H:M:S
- Year or date
- uid, gid
- Kernel version
- 32- or 64-bit kernel
- shell
- umask
- CPU type
- filesystem
- *path?*

## Reproducible FreeBSD

- Base
- Ports
- *Doc*

## FreeBSD base

- Under our control
- Almost done
  - One material open issue affecting FreeBSD-update
  - One material open issue affecting pkg base
  - ReproducibleBuilds wiki page created in 2013
  - Prompted by FreeBSD-update

# FreeBSD base - fixed

- Build date in `/usr/include/osreldate.h`
- Build host and user in `/usr/sbin/amd`
- Build date and time in `/usr/sbin/bhyve`
- Build date and time in `/etc/mail/*.cf`
- Build date in `/usr/share/doc/psd/13.rcs/paper.ascii.gz`

## FreeBSD base - fixed

Build host, user, path and time in `/var/db/mergemaster.mtree`

```
#          user: emaste
#       machine: example
#          tree: /var/tmp/temproot.fFki3iM9
#          date: Sun Apr 10 12:19:52 2016

# .
/set type=file
.               type=dir
    aliases     type=link
    amd.map     size=208 md5digest=e24ec9e1b9da870742a17669e69309a6
    apmd.conf   size=1233 md5digest=ad61867e7088f15356ce7a123b909859
    auth.conf   size=230 md5digest=5ced0a5986b19b6e60dcf25ca6d860b0
    crontab     size=723 md5digest=26d10036869afb3fd0569d9c09d44c4c
    csh.cshrc   size=106 md5digest=0fb9d8e625dcdaa81f70ee308c8135d6
    ...
```

## FreeBSD base - fixed

Build user, date and time in `/boot/loader/loader` and other loaders

```
BTX loader 1.00  BTX version is 1.02
Consoles: internal video/keyboard
BIOS drive C: is disk0
BIOS 638kB/1046464KB available memory

FreeBSD/i386 bootstrap loader, Revision 1.1
(root@logan.cse.buffalo.edu, Thu Jan  1 09:55:10 UTC 2009)
Loading /boot/defaults/loader.conf
```

*Currently requires non-default setting*

# FreeBSD base - fixed

Build user, date and time in `/boot/kernel/kernel`

```
% uname -v
FreeBSD 9.1-RELEASE #0 r243825: Tue Dec  4 09:23:10 UTC 2012
root at farrell.cse.buffalo.edu:/usr/obj/usr/src/sys/GENERIC
```

*Currently requires non-default setting*

## FreeBSD base - fixed

- Date or other metadata in filesystem image produced by makefs in `/tests/sys/geom/class/uzip`
- makewhatis output depends on man page device and inode numbers

# FreeBSD base - TODO

- Full paths in non-debug sections in kernel modules
- Host, not in-tree, makewhatis used by FreeBSD-update builds

## FreeBSD ports

- Ports do not enforce build environment (user, host name, path, ...)
- Variation good for identifying nonreproducibility
- We want to facilitate reproducibility
- Poudrière

## FreeBSD packages

- PortsReproducibleBuilds wiki page created in March 2015 (swills@)
- Initial patch sets stage dir timestamps to that of the newest distfile
- Builds vary the hostname, time, and date
- 15164 of 23599 packages reproducible (64.25%)

## FreeBSD packages

- Second iteration by bapt@
- Record timestamp in `make makesum` during port update
    - Ports r415078 by emaste@
- Use timestamp for pkg archive metadata
- Use timestamp as `SOURCE_DATE_EPOCH` in build environment

# Use timestamp for pkg archive metadata

| | |
|---|---|
| Non-reproducible | 5162 |
| Reproducible | 20009 |
| Failed | 4 |

- Example non-reproducible packages:
  GraphicsMagick, R-* *(181)*, ansible, apache-openoffice, apache24, aspell, autoconf, automake, avrdude, bind910, busybox, clamav, cmake, couchdb, courier, crashmail, cyrus-imapd25, dbus, distcc, dpkg, elixir-* *(61)*, erlang-* *(63)*, exim, fpc-* *(90)*, ficl, firefox, gcc, git, go, hadoop2, inkscape, kBuild, kde-runtime, lastpass-cli, libav, libdjbdns, libgcrypt, libgpg-error, libidn, liblz4, libtool, libxul, llvm38, m4, mongodb, node, octave-* *(91)*, openjdk, openldap-server, openvpn, owncloudclient, p5-* *(747)*, php56, py27-* *(195)*, python27, python34, qemu, ruby, rubygem-* *(1175)*, samba44, squid, subversion, tcl86, tex-luatex, thunderbird, u-boot-* *(11)*, valgrind, virtualbox-ose, vlc, wine, yacc, yasm

## Set SOURCE_DATE_EPOCH in build environment

| | |
|---|---|
| Non-reproducible | 3534 |
| Reproducible | 5062 |
| Failed | 4 |

- Example packages of 116 more now reproducible:
  calibre, cherivis-devel, cmake, easydiff, freetar, gforth, gnumail, gnustep-wrapper, kbreakout, net-snmp, mongodb32, terminal.app

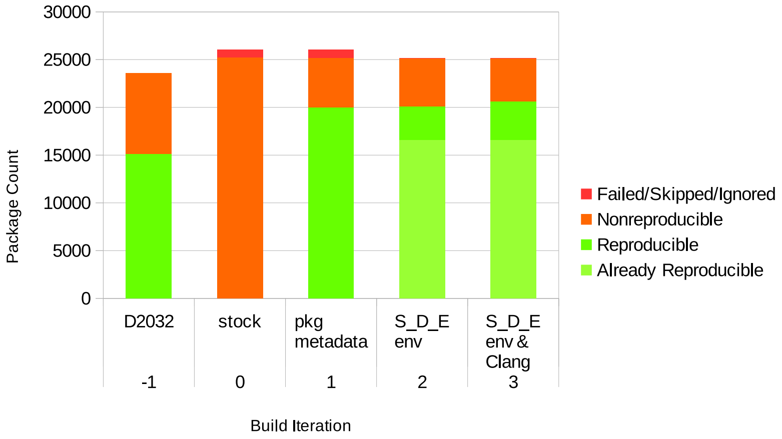
## Set SOURCE_DATE_EPOCH in build environment + Clang patch

| | |
|---|---|
| Non-reproducible | 4011 |
| Reproducible | 4549 |
| Failed | 5 |

- Example packages of 514 more now reproducible:
  abiword, analog, apache22, audacity, avrdude, bind99, bind910, clamav, crashmail, ctags, distcc, efax, exim, inkscape, libcaca, liblz4, llvm-cheri128, nagios4, nasm, rrdtool, subversion, x264, xchat, unzip

|  | D2032 | Stock | SOURCE_DATE_EPOCH pkg | +build env | +Clang |
|---|---|---|---|---|---|
| Non-reproducible | 15164 | 25222 | 5162 | 3534 | 4011 |
| Reproducible | 8435 | 0 | 20009 | 5062 | 4549 |
| Failed |  | 824 | 875 | 4 | 5 |
| Reproducible % | 64.26% | 0% | 79.49% | 79.89% | 81.92% |

# FreeBSD packages



## Investigating Changes

## What changed?

- (1) `nginx-1.10.0_3,2.txz bb31ba88b568...`
- (1) `avrdude-6.1_1.txz 0a3811a78a03...`

## What changed?

- (1) `nginx-1.10.0_3,2.txz bb31ba88b568...`
- (2) `nginx-1.10.0_3,2.txz bb31ba88b568...`

- (1) `avrdude-6.1_1.txz 0a3811a78a03...`

# What changed?

- (1) `nginx-1.10.0_3,2.txz bb31ba88b568...`
- (2) `nginx-1.10.0_3,2.txz bb31ba88b568...`

- (1) `avrdude-6.1_1.txz 0a3811a78a03...`
- (2) `avrdude-6.1_1.txz 7336a6d85dd6...`

## Diffoscope

- Examine differences *in depth*
- Output HTML or plain text
- Recursively unpack archives
- Human readable output
    - Uncompress `.PDF`, disassemble binaries
- https://diffoscope.org
- FreeBSD port `sysutils/diffoscope`
- Online version https://try.diffoscope.org

# Diffoscope

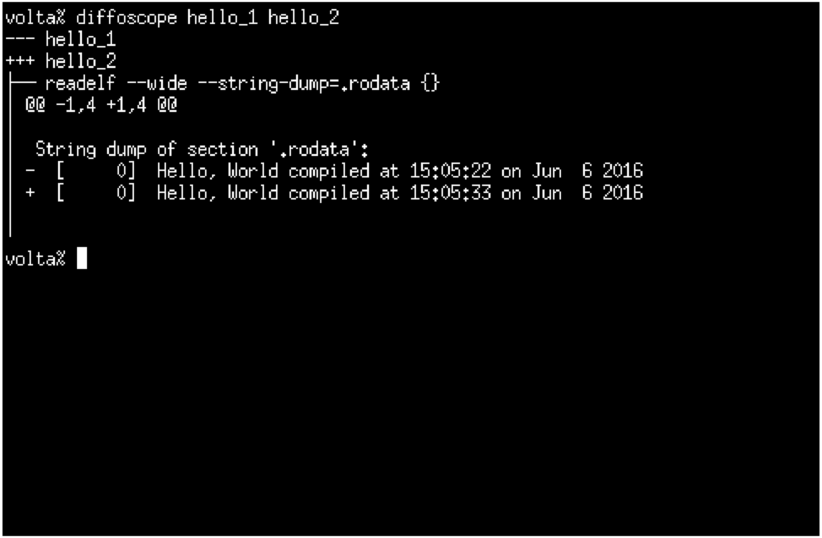
- Archives
  - bzip2, `.cpio`, `.deb`, gzip, `.ipk`, iso9660, RPM, squashfs, `.tar`, `.xz`, `.zip`
- Formats
  - Debian `.changes`, TrueType & OpenType fonts, gettext `.mo`, `.class`, Mono `.exe`, PDF, PNG, sqlite3 databases, text files
- *Contributions welcome!*

## Nonreproducible binary

### Example

```
% cat hello.c
#include <stdio.h>

int
main(int argc, char *argv[])
{
        puts("Hello, World compiled at "
             __TIME__ " on " __DATE__ "\n");
}
% cc -o hello_1 hello.c
% cc -o hello_2 hello.c
```

## Diffoscope output



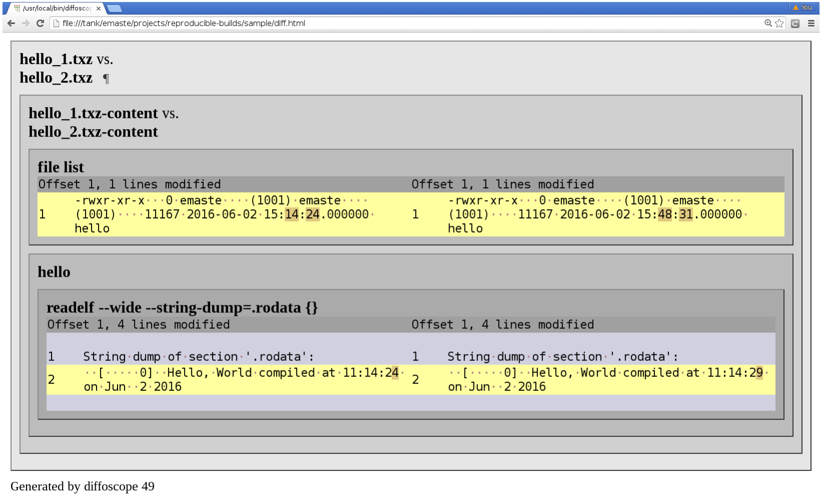
```
volta% diffoscope hello_1 hello_2
--- hello_1
+++ hello_2
├── readelf --wide --string-dump=.rodata {}
│ @@ -1,4 +1,4 @@
│
│   String dump of section '.rodata':
│ -  [     0]  Hello, World compiled at 15:05:22 on Jun  6 2016
│ +  [     0]  Hello, World compiled at 15:05:33 on Jun  6 2016
│
│
volta% █
```

## Diffoscope archives

### Example

```
% cc -o hello hello.c
% tar -cJxf hello_1.txz hello
% cc -o hello hello.c
% tar -cJxf hello_2.txz hello
```

# Diffoscope HTML output



/usr/local/bin/diffoscop
You
file:///tank/emaste/projects/reproducible-builds/sample/diff.html

**hello_1.txz** vs.
**hello_2.txz** ¶

**hello_1.txz-content** vs.
**hello_2.txz-content**

**file list**

```
Offset 1, 1 lines modified                            Offset 1, 1 lines modified
1     -rwxr-xr-x···0·emaste····(1001)·emaste····      1     -rwxr-xr-x···0·emaste····(1001)·emaste····
      (1001)····11167·2016-06-02·15:14:24.000000·           (1001)····11167·2016-06-02·15:48:31.000000·
      hello                                                 hello
```

**hello**

**readelf --wide --string-dump=.rodata {}**

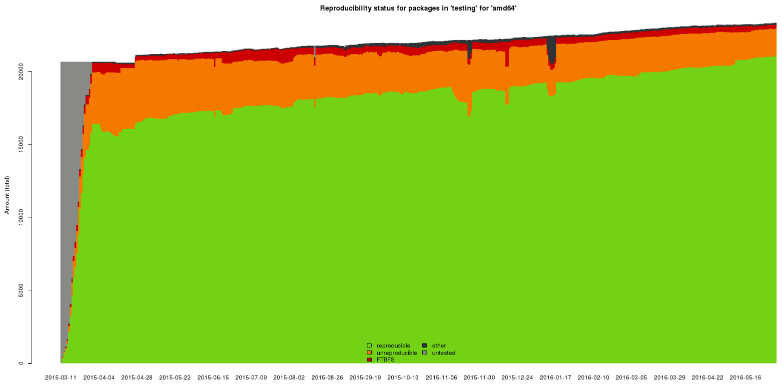
```
Offset 1, 4 lines modified                            Offset 1, 4 lines modified

1     String·dump·of·section·'.rodata':               1     String·dump·of·section·'.rodata':
2     ··[·····0]··Hello,·World·compiled·at·11:14:24·   2     ··[·····0]··Hello,·World·compiled·at·11:14:29·
      on·Jun··2·2016                                        on·Jun··2·2016
```

Generated by diffoscope 49

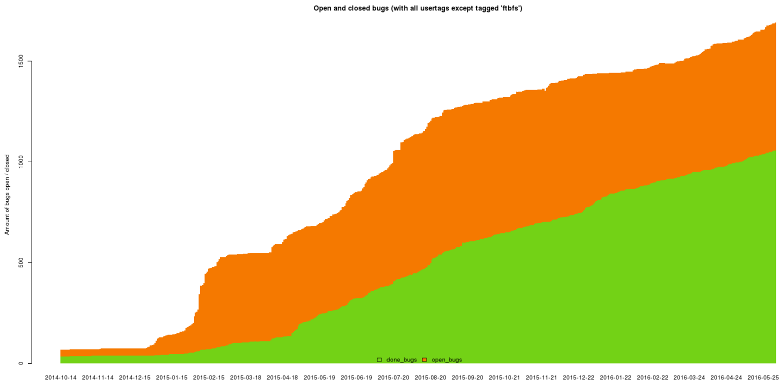
## Diffoscope

- Diffoscope is a **debugging** / **diagnostic** tool
- "Reproducible" means bit-for-bit identical
- `cmp` / `diff` / `sha256` test for reproducibility

# Debian's Experience - CI



# Debian's Experience - Bugs



## Debian's Experience - Buy-in

- Provide good arguments on why reproducible builds matter
  - ROI on security-related work not always apparent
- Continuous integration to track progress
- Reproducibility bugs come with patches

## Debian's Experience - Policy

- Debian Policy Manual
  - Structure and contents of the Debian archive
  - Design issues of the operating system
  - Technical requirements packages must satisfy to be included
- Proposed section:
  "Source must build in a reproducible manner"

## OK - Now what?

- Builders / Rebuilders
- Distribution of reproduction results
- User interface

## Thank you

## Links

- Reproducible Builds https://reproducible-builds.org/
- Diffoscope https://diffoscope.org/
- FreeBSD Reproducible Builds wiki
  https://wiki.freebsd.org/ReproducibleBuilds
- FreeBSD Reproducible Ports wiki
  https://wiki.freebsd.org/PortsReproducibleBuilds
- Debian Reproducible Builds wiki
  https://wiki.debian.org/ReproducibleBuilds
- Diverse Double-Compilation
  http://www.dwheeler.com/trusting-trust/