Reproducible Builds Summit II

December 13-15, 2016. Berlin, Germany

Aspiration, 2973 16th Street, Suite 300, San Francisco, CA 94103
Phone: (415) 839-6456 • info@aspirationtech.org • aspirationtech.org

Table of Contents

Introduction

By its very nature, free and open software allows anyone to inspect the source code for malicious flaws. However, this is not sufficient to assure that the binary machine code being executed matches the source code being inspected.

Since 2014, researches and attack reports have demonstrated how malicious code introduced into developer environments can compromise binary code while it is written by the compiler, without leaving any traces in the source code. Reproducible builds are, to the best of our knowledge, currently the only way to detect such attacks early (see documentation at https://reproducible-builds.org/docs/buy-in).

Reproducible builds allow verification that no flaws have been introduced during the compilation process, by promising identical binary packages are always generated from a given source.

As stated by Chris Lamb in the introduction to his talk at FOSSASIA 2016, entitled *Reproducible Builds – fulfilling the original promise of free software* (see http://2016.fossasia.org/schedule/#LX-01):

> “This prevents against the installation of backdoor-introducing malware on developers' machines – an attacker would need to simultaneously infect all developers attempting to reproduce the build.
>
> Furthermore, a reproducible build has a wide variety of technical advantages, including implicitly removing non-deterministic or unsafe behavior (such as downloading third-party code from the internet), detecting corrupted build environments, reducing time-to-detection of a compromised build host, as well as numerous other debugging and testing advantages.
>
> In this sense, reproducible builds are finally fulfilling the original promise of free software – that you have actual control over what is being run on your computers”.

Today, a growing number of technology projects are dedicating efforts and resources toward achieving reproducible builds, both internally and across different projects.

It was with the aim to support this thriving community to build solid and shared foundations, make critical research and development progress, and set the basis for further coordination and cooperation, that Debian, Open Technology Fund, Linux Foundation Core Infrastructure Initiative, Google Open Source Research, and Aspiration convened the Reproducible Builds Summit II.

The 3-day event took place in Berlin, Germany, from December 13 to 15, 2016, and welcomed developers from 19 projects involved in the reproducible builds community.

Summary

This report documents the work done and outcomes achieved during the Reproducible Builds Summit II.

It is structured in the following sections:

- *State of the field* provides a summary of the most notable achievements coming out of the first Reproducible Builds Summit, as well as outcomes realized by the broader reproducible builds community to date, along with a list of upcoming projects;
- *Event overview*, *Participating projects*, and *Proceedings* offer an overview of the event, including goals, an enumeration of participating projects, and a summary of scope and outcomes from the working sessions which took place;
- *Outcomes*, *Next steps*, and *Recommendations* highlight immediate outcomes of the Summit, plans within the community for the year ahead, and recommendations for promising support and investment opportunities.

State of the field

The field of reproducible builds is relatively young, but much has been achieved in a few short years. What follows is an overview of activities and progress to date, both following from the first Summit as well as across the broader community.

Notable outcomes following the first Reproducible Builds Summit

The first Reproducible Builds Summit was held in Athens, Greece in December 2015 and generated a rich array of action items and activities to focus on in the short and long term.

Among the outcomes achieved following the first Summit:

- A growing and thriving community of technology projects working on reproducible builds;
- Improved collaboration across projects;
- Confirmed plans to host two further in person meetings (one in December 2016 and one during the first half of 2017);
- A greatly improved version of the diffoscope[1] tool;
- A much clearer and agreed understanding of what buildinfo files should contain, what they mean, and how they should be used to further the goal of reproducible builds;
- For Debian: a clear view for Stretch (to be released in 2017) and Buster (to be released in 2019);
- For RPM[2] maintainers, initial understanding on how to create reproducible RPMs.

Additional progress by the reproducible builds community

In addition to the above, further progress has been made in the broader reproducible builds community.

The following are some of the most notable advancements fulfilled in this regard.

- An increasingly widespread acceptance of the concept of reproducible builds across the broader free software community;
- A growing number of public talks given at key international events, such as FOSDEM 2016, BSDCan 2016, FOSSASIA 2016 and DebConf 2016. See https://reproducible-builds.org/resources;
- Increasing positive media coverage of the community's achievements;
- An active communications set up including:
  - Website (https://reproducible-builds.org);

1 https://diffoscope.org/
2 https://en.wikipedia.org/wiki/RPM_Package_Manager

    - Discussion lists (https://lists.reproducible-builds.org/listinfo/rb-general, https://lists.reproducible-builds.org/listinfo/diffoscope);
    - Internet Relay Chat (IRC) channel (#reproducible-builds on irc.oftc.net), also used to host monthly meetings;
    - Weekly reports (https://reproducible-builds.org/news/2016/04/30/weekly-news);
    - Twitter account https://twitter.com/ReproBuilds.
- More technology projects joining the reproducible builds community: notable examples are Tails and FreeBSD;
- A Reproducible Debian Hackathon held in Boston, USA on December 3-4, 2016;
- All the most relevant development improvements available in Debian, with 92% reproducible in Debian (2015: 83%, 2014: ~60%) when building in same build path;
- Fully reproducible Bitcoin, Tor, coreboot, ElectroBSD, Webconverger;
- Inclusion of dpkg+toolchain in Debian main;
- 500 patches merged in Debian, 100+ in openSUSE;
- The creation of and ongoing work on the site https://tests.reproducible-builds.org/debian/reproducible.html, showing the potential of reproducible builds of Debian packages;
- Removal of variations related to build path from compiled binaries, including:
    - Push toward the use of the gcc's flag `-fdebug-prefix-map` (which is now in Debian's default build flags) to remove the build path from the DW_AT_producer field of the binaries;
    - Discussion and design of a new environment variable SOURCE_PREFIX_MAP similar to SOURCE_DATE_EPOCH;
    - Written patches for GCC to support all of these (and others, like stabilizing the `__FILE__` macro);
- Acceptance of --clamp-mtime into GNU tar (and also other tools), along with proposals on specifying this more formally;
- Continued development of diffoscope, including notable UI improvements (dynamic loading of large diffs, progress bar, diff folding, colors) and performance improvements;
- Creation and development of reprotest: making it work for both arbitrary build commands, and common package formats (only .dsc so far), supporting testing inside a virtual environment like a Debian schroot;
- Getting Debian buildinfo patches merged into dpkg. Now every build of a Debian package generates a buildinfo file by default;
- Design and discussion of how to serve buildinfo files from the Debian FTP archive;

- Design and discussion of the surrounding ecosystem of producers/consumers of buildinfo files;
- https://buildinfo.debian.net as an initial implementation of a buildinfo distribution service.

Current work in progress

The following efforts are currently under way in the reproducible builds community.

- A final definition of R*eproducible Build*;
- New talks and presentations;
- New communications assets, e.g. a reproducible builds logo;
- The widened range of development use cases, to be used for advocacy purposes;
- Reproducible builds documentation;
- Reproducible builds for NetBSD, Arch Linux, openSUSE, FreeBSD, Fedora, LEDE, Debian, F-Droid;
- SOURCE_PREFIX_MAP specification / GCC build path patches;
- Further diffoscope improvements;
- Buildinfo for RPM, FreeBSD, and coreboot;
- Buildinfo distribution for DAK (the Debian archive software);
- General buildinfo spec;
- RPM support;
- https://tests.reproducible-builds.org for everything non Debian;
- Reproducible live-media, containers, other images;
- Reproducible builds documentation;
- GNU coding guidelines / Debian Policy;
- 800 patches unmerged (in Debian);
- 77% reproducible in Debian (with build-path issue), 23% = 2000-5000 source packages missing.

Upcoming efforts, now in planning stage

The work on the following subjects has not started yet, but is either in the planning phase or already in the pipeline.

- A Reproducible Builds 101 publication, along the lines of "Everything you wanted to know about reproducible builds but were too afraid to ask";
- Research on policy and laws regarding reproducible builds;
- Formal Request for Comments (RFC);

- Investigating fundamentals of an executable reproducible builds description format, and tools to support this;
- Development of user controls and user verification;
- Fixing Gettext to enable reproducible translated content;
- Reproducible builds for Android, OpenBSD, Ubuntu, Red Hat;
- Buildinfo file distribution (putting the files on mirrors/blockchain/public-signed-logs);
- Enhanced analysis of buildinfo files (detect dirty build environments, find differing binaries, find same binaries even though differing environments);
- Compare https://tests.reproducible-builds.org to the “real world” (i.e. what is distributed to users);
- Compiler bootstrapping, to address the problem that compilers are themselves not reproducible;
- Cross distro issue-notes;
- pkgsrc (NetBSD & more);
- Reproducible builds and Internet of Things (e.g. for software running in cars, home appliances, etc.);
- Making MacOS and iOS builds reproducible.

Event overview

Reproducible Builds Summit II was convened at Säälchen, in Berlin, Germany from December 13 to 15, 2016.

Goals

To build on the first Reproducible Builds Summit, the main goal of the event was to further support the growing reproducible builds community, creating a space to facilitate and strengthen coordination and collaboration opportunities between projects.

Stated working goals of the meeting were:

- Update and exchange about the status of reproducible builds in various projects;
- Improve collaboration both between and within projects;
- Expand the scope and reach of reproducible builds to more projects;
- Work together and hack on solutions;
- Brainstorm designs on tools enabling end users to get the most benefits from reproducible builds.

Event program

The Summit was structured as a 3-day event, including group brainstorming, working sessions and hacking sessions.

The agenda was designed to:

- Facilitate skill and knowledge sharing;
- Encourage collaboration and coordination across shared research and development efforts;
- Build shared understanding around cross projects statements;
- Develop a strategic community-wide approach to challenges and opportunities lying ahead.

The event was documented in real time, and both agenda and session notes were stored in online etherpads openly accessible to all participants.

The session notes have been published on the reproducible builds website[3].

3 https://reproducible-builds.org/events/berlin2016/agenda/

Projects participating in the event

The following projects were represented at the Reproducible Builds Summit II in Berlin.

Arch Linux

https://www.archlinux.org

Arch Linux is a lightweight and flexible Linux distribution.

Bazel

https://www.bazel.io

Bazel is Google's own build tool. It has built-in support for building both client and server software, including client applications for both Android and iOS platforms.

coreboot

https://www.coreboot.org

coreboot is an extended firmware platform that delivers a lightning fast and secure boot experience on modern computers and embedded systems.

Debian

https://www.debian.org

Debian is a free operating system, coming with over 50000 binary packages and precompiled software bundled up for easy installation.

F-Droid

https://f-droid.org

F-Droid is an installable catalogue of free and open source software applications for the Android platform.

Fedora

https://getfedora.org

Fedora is an operating system based on the Linux kernel.

FreeBSD

https://www.freebsd.org

FreeBSD is an operating system for a variety of platforms. It is derived from BSD, a Unix operating system derivative.

GNU Guix

https://www.gnu.org/software/guix

The Guix System Distribution (GuixSD) and the GNU Guix package manager are free software projects developed under the umbrella of the GNU Project.

LEDE Project

https://lede-project.org

The Linux Embedded Development Environment (LEDE) Project is a Linux-based, embedded meta-distribution based on OpenWrt, targeting a wide range of wireless SOHO routers and non-network devices.

MacPorts Project

https://www.macports.org

The MacPorts Project is an open-source community initiative to design an easy-to-use system for compiling, installing, and upgrading either command-line, X11 or Aqua based open-source software on the OS X operating system.

NetBSD

https://www.netbsd.org

NetBSD is a free Unix-like open source operating system.

NixOS

https://nixos.org

NixOS is a Linux distribution built on top of the Nix package manager.

openSUSE

https://www.opensuse.org

openSUSE is a Linux-based project and distribution.

OpenWrt

https://openwrt.org

OpenWrt is a Linux distribution for embedded devices.

Qubes OS

https://www.qubes-os.org

Qubes OS is a security-oriented open source operating system.

Repeatr

http://repeatr.io

Repeatr assembles environments reproducibly to pave the way for reproducible use of other tools, with the ability to easily pull specific, pinned data from many different storage systems and using containers to isolate effects.

Tails

https://tails.boum.org

Tails is a live operating system that can be started on almost any computer from a DVD, USB stick, or SD card.

Tor Project

https://www.torproject.org

Tor is free software and an open network that helps you defend against traffic analysis.

Ubuntu

https://www.ubuntu.com

Ubuntu is a Debian-based Linux operating system for servers, personal computers, tablets and smartphones.

Proceedings

The Summit was facilitated and co-organized by Allen Gunn and Beatrice Martini of Aspiration, in collaboration with Holger Levsen, Chris Lamb, Ximin Luo, Mattia Rizzolo and Vagrant Cascadian from the Debian project.

A complete agenda is included as an appendix to this report.

The session notes have been published on the reproducible builds website at https://reproducible-builds.org/events/berlin2016/agenda/.

Day 1 – Tuesday, December 1

Opening session

The Summit opened with words of welcome from the hosts and appreciation to the partners and co-organizers who supported its organization.

Participants introduced themselves to the group and the facilitator provided a brief overview of meeting format and participant guidelines.

The first session of the morning was a project showcase.

Project showcase

This session was designed to inform those in attendance about the outcomes achieved at the first Reproducible Builds Summit and discover more about the progress made to date.

Nine participants, each of them representing a project involved in the reproducible builds community, hosted small-group Q&A conversations. The other participants were invited to join a conversation of their choice, and every few minutes had the opportunity to visit a new group.

The following is an overview of the updates shared by the projects presenting at the showcase.

Reproducible FreeBSD

Ed Maste

FreeBSD is an advanced computer operating system used to power modern servers, desktops, and embedded platforms.

Reproducibility efforts in FreeBSD started several years ago on a somewhat ad-hoc basis, but over the last year build reproducibility has become a topic of greatly increasing interest, with a combination of efforts from upstream software developers and open source operating system developers and packagers. There are many reasons why software does not build reproducibly, including timestamps embedded in object files, timezone and locale settings affecting the build, output that depends on the order in which files are returned by the file system, and metadata stored in archive files.

Further details about the current state of build reproducibility on FreeBSD are discussed in Ed Maste's talk at BSDCan 2016, archived at https://www.bsdcan.org/2016/schedule/events/714.en.html.

Reproducible coreboot, OpenWrt, LEDE

Alexander Couzens (lynxis)

Coreboot is an extended firmware platform that delivers a fast and secure boot experience on modern computers and embedded systems.

Reproducible coreboot is an effort to apply reproducible builds to coreboot. Thus each coreboot.rom is built twice (without payloads), with a few variations added and then those two ROMs are compared using diffoscope. Further details are published at: https://tests.reproducible-builds.org/coreboot/coreboot.html.

OpenWrt is described as a Linux distribution for embedded devices. Instead of trying to create a single, static firmware, OpenWrt provides a fully writable filesystem with package management.

Reproducible OpenWrt is an effort to apply reproducible builds to OpenWrt. Thus each OpenWrt target is built twice, with a few variations added and then the resulting images and packages from the two builds are compared using diffoscope. OpenWRT generates many different types of raw .bin files, and diffoscope does not know how to parse these. Thus the resulting diffoscope output is not nearly as clear as it could be. Hopefully this limitation will be overcome eventually, but in the meanwhile the input components (uImage kernel file, rootfs.tar.gz, and/or rootfs squashfs) can be inspected. More updates can be found at: https://tests.reproducible-builds.org/openwrt/openwrt.html.

The LEDE Project is a Linux-based, embedded meta-distribution based on OpenWrt, targeting a wide range of wireless SOHO routers and non-network devices. LEDE is an acronym for "Linux Embedded Development Environment".

Reproducible LEDE is an effort to apply reproducible builds to LEDE. Its documentation is online at https://tests.reproducible-builds.org/lede/lede.html.

Building reproducible Tails ISO images

intrigeri

Tails is a live operating system that you can start on almost any computer from a DVD, USB stick, or SD card.

Information about the motivations to apply reproducible builds to Tails can be found at https://tails.boum.org/blueprint/reproducible_builds/#why.

The Tails team wants to provide a verifiable path from human readable Tails source code to the binary ISO images we publish. In other words, Tails wants to enable anyone (given sufficient technical skills and hardware resources) to rebuild from source a given Tails release, in order

to independently verify that it matches the ISO image we have published. Further details are published at https://tails.boum.org/blueprint/reproducible_builds/#how.

RPM and reproducible builds in openSUSE

Bernhard Wiedemann

OpenSUSE is a Linux-based project and distribution.

The efforts made to apply reproducible builds to openSUSE highlighted challenges regarding embedded timestamps, hostname, embedded rebuild counters, variations in .o files due to random link order, compile-time optimization and CPU detection.

Scripts to help make openSUSE builds reproducible can be found at https://github.com/bmwiedemann/reproducibleopensuse.

These tools are intended to make it easier to verify that the binaries resulting from openSUSE builds are reproducible. In order to achieve this, it is necessary to rebuild locally from source and compare the results with published binaries using the build-compare script that abstracts away some unavoidable (or unimportant) differences like RPM timestamps. Ultimately, the goal is to dispose of that script too.

F-Droid and reproducible Android apps

Hans-Christoph Steiner

F-Droid is an installable catalogue of FOSS applications for the Android platform. The client makes it easy to browse, install, and keep track of updates on your device.

F-Droid has been working towards getting a complete app distribution channel that is able to reproducibly build each Android app from source. This provides tangible benefits. First, it means that anyone can verify that the app that they are using is 100% built from the source code, with nothing else added. That verifies that the app is indeed 100% free, open source software. It also verifies that there have not been any malicious bits of code added into the app during the build process.

Some quick checks show that a large number of the apps in f-droid.org already build reproducibly, given the right build environment. F-Droid is working on making the process of setting up that build environment as automated as possible.

More information about F-Droid's reproducible builds work is documented in their blog updates archive: https://guardianproject.info/tag/reproducible-build.

Eliminating absolute build paths from debugging info

Ximin Luo

As documented on the reproducible builds Debian blog on November 7, 2016 (online at https://reproducible.alioth.debian.org/blog/posts/80, linking to a more detailed thread at https://gcc.gnu.org/ml/gcc-patches/2016-11/msg00182.html):

“[Debian is working on making] build processes be able to reproduce the build outputs independently of which filesystem path the build is being executed from – e.g. if the executing user doesn't have root access to be able to execute it under a standard path such as /build. This currently is making about 2k-3k [...] packages in Debian unreproducible when build-paths are varied across builds. [,,,]

So, we believe it is better to patch GCC centrally. Our proposed solution is similar to (a) the SOURCE_DATE_EPOCH environment variable which was previously accepted into GCC and was used to successfully make 400+ packages reproducible, and (b) the -fdebug-prefix-map mechanism that already exists in GCC and which nearly but not quite, achieves at-scale build-path-independent reproducibility”.

Tests.reproducible-builds.org. How we constantly test Debian

Mattia Rizzolo

The website https://tests.reproducible-builds.org/debian/reproducible.html demonstrates the potential of reproducible builds of Debian packages. The results shown were obtained by several jobs running on jenkins.debian.net.

Besides Debian, also coreboot, OpenWrt, NetBSD, FreeBSD, Arch Linux and LEDE, are being tested on https://tests.reproducible-builds.org. Testing of Fedora has just begun, and there are plans to test F-Droid and GNU Guix.

The website administrators are reachable via IRC (#debian-reproducible and #reproducible-builds on OFTC), or email (qa-jenkins-dev@lists.alioth.debian.org).

Feedback and collaboration proposals from upstream developers or from anyone working on another distribution is very much welcome.

Buildinfo.debian.net. An experiment in distributing build information at scale

Chris Lamb

In order to build packages reproducibly, both identical sources and some external definition of the environment used for a particular build are needed. This definition includes the inputs and the outputs and—in the Debian case—are available in a $package_$version_$architecture.buildinfo file.

It is not currently clear how these files could or should be handled in practice, hence the creation of this server to investigate.

Reproducible Debian status

Holger Levsen

As reported on the reproducible builds Debian wiki: “Reproducible builds of Debian as a whole are still not a reality, though individual reproducible builds of packages are possible. So while we are making very good progress, it is a stretch to say that Debian is reproducible”.

Further details about current status and upcoming next steps can be found at https://wiki.debian.org/ReproducibleBuilds.

Information about how to contribute and be in the loop are listed at https://wiki.debian.org/ReproducibleBuilds/Contribute.

Agenda brainstorming

After the project showcase, participants were invited to articulate which, in their opinion, were the most critical topics to be addressed during the Summit. Expressed as a statement or question, each topic was to be written on a post-it note, allowing then for visualization and subsequent categorization of the outputs.

Once organized and displayed on the wall, the emerging themes offered the opportunity to prioritize topics for subsequent discussion and collaboration.

The resulting collection of data provided with a compelling snapshot of the group's current perspective on the reproducible builds work.

The following listing represents the categories which came into view from the examination of the session's results.

- Definition of Reproducible Build
- Advocacy to [other technology] projects
- Outreach and communications (non technical people)
- Outreach and communications (developers)
- Project status
- Docs [documentation]
- Compilers
- Tool support
- build info (general, doc, advocacy)
- build info (implementation) / Debian infrastructure
- Build paths / SOURCE_PREFIX_MAP
- SOURCE_DATE_EPOCH
- SquashFS
- Reproducible file systems + images (see also: SquashFS)
- RPM
- Source code
- diffoscope

- Skill share [proposals and requests for the upcoming Summit Skill Share session]
- Legal and law
- Binary transparency
- User verification
- Tests.reproducible-builds.org: generic
- Tests.reproducible-builds.org: cross distro
- Reprotest
- Cross distro
- Bootstrapping
- Funding

Working sessions I

The afternoon was dedicated to the first round of working sessions. The following is a brief summary of the sessions which took place.

Diffoscope: Current status, challenges and next steps

Chris Lamb

Session participants focused on reviewing the inputs regarding diffoscope collected through the Agenda Brainstorming session.

The discussion focused on:

- How the platform could be improved, for example with further support for distro specific or uncommon file formats;
- The integration of debdiff and diffoscope;
- Parallel diffoscope;
- Diffoscope plugins;
- Usability issues;
- Automatic classification of reproducibility issues in diffoscope.

The session identified the following action items:

- Open a bug for the output format accessibility;
- Incorporate two patches produced by FreeBSD to improve portability
  - patch 1: https://anonscm.debian.org/git/reproducible/diffoscope.git/commit/?id=6812c22596d531df18fffe43bfb7c774f2d09307

- patch 2: https://anonscm.debian.org/git/reproducible/diffoscope.git/commit/?id=fec9e97c51b3a8ff226a4b3b2b0563a4a680ac68.

Reprotest: Project overview and potential improvements

Ximin Luo

Reprotest is an all-in-one tool allowing anyone to check whether a build is reproducible or not, replacing the string of ad-hoc scripts the reproducible builds team used so far.

Projects outside Debian have set up their own tools to check reproducibility. The idea of reprotest is to have a general tool for everyone.

Its usability is not optimal yet: how could it be improved? It would be useful to make a list of all the variations that any tool could make to check reproducibility, and then when a package is not reproducible, bisect the variations to figure out which one makes the build different.

Follow-up action emerged from the session: build a package twice repeating the same environment as close as possible, and then start adding variations to figure out which one causes a difference in the build.

How to improve the reproducible builds documentation

Holger Levsen

The efforts of the reproducible builds community were initially documented on the reproducible builds Debian wiki (https://wiki.debian.org/ReproducibleBuilds) and later started to appear more regularly on the reproducible builds website (https://reproducible-builds.org).

The group discussed how to permanently organize all the community's documentation on the website and how to improve content and accessibility of the documentation.

Among the examples of new content discussed by the group:

- A clear definition of reproducible builds;
- Concise FAQs able to guide readers depending on their interests;
- Examples of projects that have successfully worked on reproducible builds.

How to enable the user verification process

Daniel Kahn Gillmor

Reproducible builds are becoming increasingly available, and it is now time to discuss how we want regular users (not just developers and advanced system administrators) to experience their benefits.

Changing package managers so that they will exclusively install builds which are reproducible seems like one likely avenue (both to improve end-user security, and to incentivize distro packages to demand reproducibility before releasing).

The group discussion highlighted the necessity to show the full records of multiple builds to the end-user's package manager, so that the package manager can locally confirm that the builds were reproducible. In fact, there is no improvement if we simply trust a single signature from an upstream claiming a package is reproducible: it is critical to see multiple signatures of independent parties who performed their own builds.

Furthermore, session participants talked about the promising opportunity for end users to be able to easily run a build themselves in case they do not have enough upstream confirmations already.

The session ended considering that future work on user verification is surely necessary to describe how we identify different builders, how we share their logs, and how we should determine which build records are appropriate to compare.

Embedded systems and reproducible builds

Alexander Couzens (lynxis)

The session focused on the challenges concerning the use of reproducible builds in the Internet of Things sector.

Among them:

- It is not yet possible to ensure that a vendor has not installed malicious firmware in a device;
- Coreboot cannot currently ship binaries.

The following proposals emerged from the discussion:

- Third party vendors should be encouraged to publish hashes of firmware shipped with hardware;
- coreboot should publish hashes for a select number of standard configurations/boards.

How to make RPM reproducible I

Dennis Gilmore, Bernhard Wiedemann

Session participants discussed a wide range of challenges currently encountered in making RPM reproducible and brainstormed opportunities for improvement.

The development of two new tools was discussed:

- A tool to reproduce the environment (build input, etc.) that would take build info and set-up the build environment;
- A tool to generate build-info from RPM file which, for example, would be of use for Qubes OS.

Closing plenary

The first day of the Summit ended with a closing plenary, during which participants were invited to share what topics they thought should be prioritized on the agenda for Day 2.

Day 2 – Wednesday, December 14

Opening session

The second day of the Summit started with an opening session that summarized the first day of the Summit and offered an overview of the Day 2 agenda.

A second round of sessions was announced, and participants joined their working group of choice.

The following are brief reports documenting the outcomes of each conversation.

Working sessions II

How do buildinfo files work

Ximin Luo

The scope of the session was to describe what buildinfo files are, what they do, and how they can help the reproducible builds work.

A buildinfo file is a record of what a particular builder did to build some binary output. It contains as much information about the build environment as is reasonable to distribute, and attempts to include all information needed to reproduce that build.

The buildinfo file has several related goals:

- It records information about the system environment used during a particular build – packages installed (toolchain, etc), system architecture, etc. This can be useful for forensics/debugging;
- It can also be used to try to recreate, partially or in full, the system environment when trying to reproduce a particular build.

Among the upcoming tasks regarding the work on buildinfo files:

- Collate and distribute buildinfo files not only from Debian developers and the official builds, but also from external parties;
- Write tools making it easy to test reproducibility and submit buildinfo;
- Write tools to retrieve buildinfo files/signatures when installing.

How to make RPM reproducible II

Dennis Gilmore, Bernhard Wiedemann

Participants discussed the specifics that would need to be implemented to enable reproducibility for RPM, and how to use these more consistently across different projects.

The discussion helped to start enumerating what steps would be needed and what current open questions the community should focus on. The following were the actionable next steps agreed on:

- Create a tool to generate buildinfo files similar to Debian's (see https://buildinfo.debian.net);
- Subsequently create or extend a tool to use buildinfo to create a similar environment to rebuild a package later.

The development work emerged from the session can be found online at https://github.com/woju/rpmbuildinfo.

How to share the work done by different projects on reproducible images

Hans-Christoph Steiner

By composing base images, it is possible to create an (almost) reproducible environment and, using an appropriate cloud service, easily deploy those images.

Several tools have been designed to help with running virtual machines and containers, some with the explicit goal of reproducible builds in mind, e.g. Docker, Vagrant, Gitian.

The underlying theme of the conversation about Docker, Vagrant, Gitian is “tools we can use to give us a reproducible environment, so that we make it natural to have executable reproduce instructions inside it”.

Session participants discussed how different projects build images and what they use for this purpose: e.g. Bazel uses Docker, F-Droid uses Vagrant.

The discussion wrapped up with a general agreement: which virtual machine or container is used does not matter for reproducibility, but it matters for ease of use and set up.

How to define a reproducible build I

Ed Maste

Participants to this session worked together to develop a collectively agreeable definition of *reproducible build*.

The group agreed on saying that a reproducible build:

- Is bit-by-bit identical;
- Is checksum verifiable;
- Has specified outputs;
- Has no specialized comparator.

The ideal reproducible build has:

- Same source code;
- Build instructions (command line);

- Same environment configuration, build flags;
- Dependencies and their versions;
- Locations where dependencies are installed.

Ideally, we only need to specify:

- Source code;
- Build instructions command line;
- Configuration, build flags;
- Dependencies and their versions.

A build is viably reproducible if the following need to be specified:

- Locations where dependencies are installed;
- Build path prefix;
- LOCALE;
- Known signatures;
- Saved optimization metadata;
- SOURCE_DATE_EPOCH.

A build is unreproducible if dependent on:

- Host-dependent optimization (save a specific one is better);
- System time (save a specific one is better. See SOURCE_EPOCH_DATE);
- Random data;
- Readdir order;
- A specific person;
- Signing keys (because that means it depends on a specific person).

Participants in the session proposed a follow-up session during the event to author a draft of the definition and share it with the wider community to invite feedback.

What end user policy should reproducible builds have?

Daniel Kahn Gillmor

The aim of the discussion was to identify the ideal user policy for an average user.

The minimum viable option: the distribution only publishes packages that meet its own internal definition of reproducibility (e.g., it builds the same way on three different build servers).

A more nuanced policy: the user designates builders they are willing to trust (initially seeded from the distro), numbered N, and requires that K of those N builders achieve the same result before they're willing to install the package.

Among the open questions identified:

- Is there a privileged buildinfo, e.g. one that other builders are expected to match?
- Is this the distro's own buildinfo?
- Can power users build the package themselves and use that to help satisfy K, or override completely?
- How should we communicate that there are dissenting results when K agree?
- What happens if a package stops meeting policy after it has been installed previously?
- How many builders can we provide per architecture?

Improving the test infrastructure

Mattia Rizzolo

The session focused how the test infrastructure works and collecting feedback on its current status.

A hefty list of action items was agreed on:

- Eliminate iframes from the tests.reproducible-builds.org test setup;
- Reproduce the test set up; merge the postgres code and create new schema;
- Document notes useful for projects other than Debian;
- Make the database schema more cross distro friendly;
- Invite developers of projects other than Debian to do reproducible tests and share their outcomes.

Fixing Gettext

John Darrington

The problem discussed was that timestamps end up in binaries because of Gettext (the issue is discussed in greater detail at https://savannah.gnu.org/bugs/?49654).

Debian implemented a patch to make xgettext respect SOURCE_DATE_EPOCH, but it was rejected by the former maintainer. Session participants came up with an alternative approach: instead of using SOURCE_DATE_EPOCH for translation templates, which may be inaccurate, they computed the latest modification time for all source files and use *that* in the timestamp instead of the current time. A patch was subsequently prepared.

Furthermore, additional patches were prepared for other approaches such as omitting the timestamp header from the .mo files. The upstream would then be able to pick from one of the possible solutions.

Skill share sessions

The morning proceedings drew to a close with a skill share session. Participants who wanted to share a skill relevant to the Summit's work were invited to host a concise Q&A conversation with one or a few more interested peers (the formation of very small groups was encouraged, in order to allow for more direct knowledge exchange).

The following is a list of the conversations which took place.

- Git-based packaging
- How to use C sanitizers and fuzzing
- How to make storage deduplicate and incentivize reproducible builds hash
- How to use buildinfos to analyze/test reproducibility
- Fedora ask-me-anything
- How to use Emacs
- How to run a start-up
- How to apply for funding from the Core Infrastructure Initiative
- How to improve cross distro packaging https://maintainer.zq1.de
- How (not) to use iframes on awesome webpages
- How to sign code in git (and correctly verify the signatures)
- Ask me anything about building on OSX
- How to do automatic hardware testing

Working sessions III

At the core of the afternoon's agenda was a third set of working sessions, featuring follow up sessions based on action items emerged during the first half of the Summit, as well as specific discussions about topics not yet thoroughly examined.

The following is an overview of the sessions which took place, also including a short summary of the discussion items they focused on.

What else for the auditable ecosystem?

Hanno Böck

Reproducible builds are just part of what allows to enable reproducibility. The session focused on discussing what else is needed.

Participants talked about the auditable trust check they would like to have. If we have verifiable source code, we can get benefits of trust in the code by running stuff that was made from it. The conversation helped identify several critical open questions:

- Does binary-transparency-style logging perform the same functionality as trusting N of K?
- Would it be possible to hire someone to do a formal security analysis of git?
- What would a "git2" look like?
- Can we get git with a different hash? Work is being done in this regard, but it is a slow process, as sha1 is kind of hardcoded everywhere in git's sources.
- Do we have a place to turn to when checking our view (or mirror) of upstream source code hasn't been tampered with? There is not.

The following was one of the most compelling realizations emerged during the session.

- Should we make some sort of "Ecosystem Janitor Team" which makes sure sources are mirrored and the hashes publicly logged? It would be great. There are things such a team could start working on right now that would benefit all distros, such as building tools to mirror and publicly log hashes of upstreams.

SOURCE_PREFIX_MAP

Reiner Herrmann

The aim of the session was to try to come up with a specification for the SOURCE_PREFIX_MAP environment variable.

To the question "Should the specification contain only one or multiple mappings?", participants answered that multiple mappings would be better, as it would allow for more intuitive overrides by child processes. Further open questions were then identified and discussed.

Among them:

- How to apply the mappings when eventually set?
- What would be the exact format of the environment variable?

Actionable next steps agreed upon:

- Research passing newlines through shell, m4, autoconf;
- Look how gdb parse and loads symbol paths to source code paths.

How to improve the reproducible builds documentation II

Valerie Young

The group talked about how to improve the https://reproducible-builds.org website.

Participants created an etherpad to invite community members to propose and answer questions they would like to have included in the FAQ page.

At the end of the session, they also asked all Summit participants for feedback regarding the idea of adding a page on the main site for each project working on reproducible builds, as a place to include the project's own relevant documentation.

How to define a reproducible build II

Ed Maste

Session participants drafted their proposal for a definition of reproducible builds.

The proposal was presented at the end of the session, discussed on the Summit mailing list afterwards, and it appeared in its first published version on the reproducible builds website on December 20 at https://reproducible-builds.org/docs/definition.

It reads as follows:

> "When is a build reproducible? 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".

Discussion is ongoing regarding refinements to this definition.

Writing a statement about what it means to do bootstrappable compilers I

Ludovic Courtès

The session aimed at making a public statement about why bootstrapping is a problem and what best practices can be recommended to compiler writers.

Participants agreed on setting up a website to host their manifesto.

The manifesto would:

- Describe what bootstrapping is;
- Draw attention to the need for an auditable, repeatable process for bootstrapping programming languages, compilers, pieces of the toolchain and whole distributions;
- Provide examples of possible benefits and suggestions regarding best practices.

Once the content is marshaled and reviewed, the website will go live at http://bootstrappable.org.

Reproducible builds use cases

Holger Levsen

A first list of reproducible builds use cases was developed during the first Reproducible Builds Summit and can be found online at https://reproducible-builds.org/docs/buy-in.

Starting from there, the session aimed at discussing further benefits making the reproducible builds work worth pursuing, to be added to the original inventory.

Among the benefits agreed upon:

- Increased development speed by limiting number of rebuilds of dependent packages, which equals lower costs;
- Increased development speed based on less QA work before release allows for better security by reduced exposure time to 0day exploits;
- QA and code quality allow for detection of issues of failing builds under rare circumstances, such as timing/races, number of cores, locales;
- Using less storage enables savings on expenses for hardware;
- If reproducible builds tools are executable, they give a clear victory condition for one-step builds (the importance of which is widely touted in the "Joel Tests" for software company quality): process quality and team productivity increases.

Using reproducible build information to help GPL compliance

Brett Smith

The purpose of the session was to discuss the relationships between licensing and reproducibility. The group talked about how reproducible builds could help prevent GPL violations. The consensus in the session was that reproducible builds held more promise to help distributors put together a source package than to fill the gap after a violation occurred.

Among the actionable next steps identified:

- Provide an SDK, including common toolchain for embedded development that generates proper reproducibility with buildinfo or related information necessary to build, and talk to linaro to ensure that their toolchains contain reproducibility information;

- Generate buildinfo files, embedding hashes of includes and other build source in binaries;
- Contact embedded distros and get openembedded, angstrom, etc. using reproducibility toolchains.
- One particular tool that could be built: a source code tarball tool that looks at a source release, and buildinfo and identifies missing pieces from the source code.

Closing plenary

The second day of the Summit wrapped up with a closing plenary which included an opportunity for participants to propose projects to work together on during the rest of the afternoon.

The following is the list of the projects and themes participants proposed to work on.

- Work on the next steps identified by the diffoscope session on Day 1
- Reproducible builds website documentation sprint
- Core Infrastructure Initiative funding opportunities
- Help make RPM reproducible
- Apply reproducible builds to Nix and test at https://tests.reproducible-builds.org
- Reproduce the https://tests.reproducible-builds.org test set up
- Embed images cross-distro

Day 3 – Thursday, December 15

Opening session

The third and final morning of the Summit started with a brief opening session, followed by a last set of working sessions and additional project hacking.

Working sessions IV

Cross distro collaboration on reproducible builds

Dennis Gilmore

Session participants discussed how to facilitate cross distro communication. Posting on the reproducible-builds-general list is ok, but many community members don't seem to know about the list and it should be more widely advertised. Also having more clearly displayed information on the website about how to get involved would help.

The main existing challenge lies in the fact that all the current reproducible builds infrastructure is hosted on Debian. In order to get involved, you need to know the Debian process: what channels to use, how to file bugs. It is a barrier to entry.

The group also talked about generalizing buildinfo tools, considering for example the opportunity to capture buildinfo for things like distro Docker images: however some participants felt like that the process would inevitably be distro specific, ultimately limiting the utility of any general-purpose tool.

Writing a statement about what it means to do bootstrappable compilers II

Ludovic Courtès

Following up on the first session focusing on this effort, the group drafted a first version of the statement, which is now live at http://bootstrappable.org.

How to improve the reproducible builds documentation III

Valerie Young

The group kept working on the list of tasks developed during the previous Documentation session. Current priorities:

- Complete the FAQ page to welcome new visitors to the site and lead them to the information they need, depending on what they want to learn about or contribute to;
- Identify different user persona and write a "Why should I care about reproducible builds?" section for each of them;
- Create a "How to start contributing" page with a clear list of contact points such as mailing lists and IRC channels

- Building on the existing content published at https://reproducible-builds.org/tools, create new pages to describe each tool more in detail, also including information like bug tracking instructions, releases list, and link to git.

A mirror of the website can be found at https://github.com/reproducible-builds, and there are plans to set up an automatic mirror of diffoscope as well.

Using the idea of "certificate transparency" for packages

Eric Myhre

Certificate transparency logs use a special cryptographic mechanism to facilitate public auditing of certificates and logs. This special cryptographic mechanism, known as a Merkle hash tree, is a simple binary tree consisting of hashed leaves and nodes.

This session focused on the following question: can we use the same idea of certificate transparency for packages?

Answering this query is critical for gaining an understanding of what the build result should be, as well as for allowing people who do not want to build from source to read who produced the binary they are trying to install.

With web site certificates, we have the information about when the certificate was issued and when it expires. But an unanswered question remains regarding what to do with the idea of revocation.

Session participants agreed on the need for further examination of different logs to collect more data to investigate. At the end of the session, they invited community members from all projects to share logs, in order to collect a relevant variety and then work on determining how to best facilitate public auditing.

State of reproducible builds

Holger Levsen

The goal of the session was to make an inventory of all the reproducible builds efforts completed, in progress, or in planning phase. The full inventory is included in this report in the State of the Field section.

Among the completed and ongoing activities:

- Communication channels setup (website, IRC channel, discussion lists);
- Monthly IRC meetings and weekly reports;
- Two Summits hosted;
- A growing developer community;
- 92% reproducible builds in Debian (2015: 83%, 2014: ~60%) when building in same buildpath;
- Fully reproducible Bitcoin, Tor, coreboot, ElectroBSD, Webconverger;

- https://tests.reproducible-builds.org for Debian;
- written specification for SOURCE_DATE_EPOCH (see https://reproducible-builds.org/specs/source-date-epoch), and patches applied to various projects (including GCC, TexLive, etc.);
- Initial prototype of reprotest; buildinfo for Debian.

Among the efforts currently in progress:

- Reproducible build definition;
- Use case inventory;
- Advocacy efforts;
- Reproducible builds for NetBSD, Arch Linux, openSUSE, FreeBSD, Fedora, CEDE, Debian, F-Droid;
- SOURCE_PREFIX_MAP specification / GCC build path patches;
- Diffoscope improvements;
- Buildinfo for RPM, FreeBSD, coreboot;
- Buildinfo distribution for DAK (the Debian archive software);
- https://tests.reproducible-builds.org for anything non-Debian;
- Reproducible builds documentation.

In the pipeline:

- Reproducible builds for Android, OpenBSD, Ubuntu, Red Hat;
- Buildinfo file distribution (putting them on mirrors/blockchain/public-signed-logs);
- Analyze buildinfo files (detect dirty build environments, find differing binaries, find same binaries even though differing environments);
- Compiler bootstrapping;
- Cross distro issue-notes.

Project hacking

The second part of the morning was dedicated to project hacking on shared efforts.

The following is a list of the projects participants proposed to work on:

- Diffoscope debug;
- RPM;
- Reproducible builds documentation;
- Buildinfo files coming from different architectures;

- Gettext;
- FreeBSD filesystem;
- Android documentation;
- Reproducing the https://tests.reproducible-builds.org test environment and documenting it;
- Bootstrapping manifesto;
- Binary transparency log;
- Android infrastructure.

Looking ahead to 2017

To conclude the Summit while looking to the future of the reproducible builds work, participants were invited to join to a final session entitled *Looking ahead to 2017*.

8 stations dedicated to 8 different domains of the reproducible builds work were hosted. All Summit participants were invited to visit them all and share two types of data:

- Goals worth achieving in that specific domain in one year;
- Unknown items or questions that need to be resolved in that same domain during the following year.

The following is the list of the stations hosted. Outcomes of the session are presented in further detail in the Next Steps section of this report.

- Advocacy/Outreach/Policy/Education
- Community/Documentation
- Coverage/Upstreams
- Hardware/Embedded
- Dev tools/Toolchain
- User Tools
- Testing Infrastructure
- New Directions

Closing plenary

Finally, the Summit ended with a closing plenary where participants highlighted accomplishments from the meeting, and shared mutual appreciation for the outcomes achieved together.

Event outcomes

The second Reproducible Builds Summit achieved a wide range of outcomes, from the development of stronger and better coordinated plans, to collaboration on shared documentation resources and the development of code committed to the different participating projects.

General outcomes

- Greater knowledge and skill exchange between members of the 19 participating projects;
- A first definition of reproducible build, now published at https://reproducible-builds.org/docs/definition;
- A statement about what it means to do bootstrappable compilers, available at https://bootstrappable.org;
- Plans, already well under way, for a thorough and collective effort to improve documentation and training materials on https://reproducible-builds.org;
- Development of new use cases to advocate for the importance of reproducible builds across technical, corporate and legal sectors.

Improved tools

- Diffoscope: a diffoscope bug (https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=848249) was fixed during the summit, and several other patches have been written and merged already;
- Reprotest: Agreement was reached on making a list of all the variations that any tool could make to check reproducibility, and then when a package is not reproducible, bisecting the variations to figure out which one makes the build different.

  Next step: building a package twice repeating the same environment as close as possible, and then adding variations to identify which component causes a difference in the build.

Patches submitted

- A number of FreeBSD changes were committed at the summit and in the following days:
  - newvers.sh: add option to eliminate kernel build metadata https://svnweb.freebsd.org/changeset/base/310112
  - Add WITH_REPRODUCIBLE_BUILD src.conf(5) knob to disable kernel metadata https://svnweb.freebsd.org/changeset/base/310128

    - Build loaders reproducibly when WITH_REPRODUCIBLE_BUILD https://svnweb.freebsd.org/changeset/base/310268
    - bhnd: remove srand() to ensure deterministic output https://svnweb.freebsd.org/changeset/base/310371
    - mlx: avoid use of __DATE__ to make build reproducible https://svnweb.freebsd.org/changeset/base/310425
- The diffoscope team merged patches to improve support for running the tool on FreeBSD. They are recorded at:
    - https://anonscm.debian.org/git/reproducible/diffoscope.git/commit/?id=6812c22596d531df18fffe43bfb7c774f2d09307
    - https://anonscm.debian.org/git/reproducible/diffoscope.git/commit/?id=fec9e97c51b3a8ff226a4b3b2b0563a4a680ac68
- The maintainers of gettext applied the patches sent to remove timestamps from the output of gettext generated .mo files
    - http://git.savannah.gnu.org/cgit/gettext.git/commit/?id=d13f165b83701dffc14f7151419e0c00c00c0d1b
- Documentation improvements were made to the reproducible builds website
    - https://lists.reproducible-builds.org/pipermail/rb-general/2016-December/000184.html
    - https://lists.reproducible-builds.org/pipermail/rb-general/2016-December/000185.html
    - https://lists.reproducible-builds.org/pipermail/rb-general/2016-December/000187.html
    - https://lists.reproducible-builds.org/pipermail/rb-general/2016-December/000190.html
    - https://lists.reproducible-builds.org/pipermail/rb-general/2016-December/000192.html

Post-event blog posts

Several participants posted reflections about the event.

- Debian reproducible builds team
    - https://reproducible.alioth.debian.org/blog/posts/86
- GuixSD team
    - https://www.gnu.org/software/guix/news/reproducible-build-summit-2nd-edition.html
- Rok Garbas
    - https://garbas.si/2016/reproducible-builds-summit-in-berlin.html
- Brett Smith, Software Freedom Conservancy
    - https://sfconservancy.org/blog/2016/dec/26/reproducible-builds-summit-report

- Clemens Lang, MacPorts Project
  - https://lists.macports.org/pipermail/macports-dev/2016-December/035052.html

Next steps

The final session of the Summit, called Looking Ahead to 2017, provided participants with an opportunity to articulate their predictions and wishes for reproducible builds efforts during the coming year.

The following are projections across 8 different domains which emerged as key priority areas for reproducible builds work in the short term. They are presented in their raw form, and will be tracked across the coming year and in subsequent reports.

Advocacy/Outreach/Policy/Education

- Documentation:
  - Teach people how to write reproducible code
  - SOURCE_PREFIX_MAP specification done
  - Document all indirect highly valuable side effects of reproducible builds
- Outreach
  - Reproducible Builds Summit III in 2017
  - More talks at conferences around the world
  - Reproducible hacking sprint 2017 (hackotlaw)
  - White papers and academic research
  - Explain reproducible builds to my grandmother/grandparent/grandchild in five sentences
  - All conferences having reproducible builds tracks
  - Start outreach to embedded device manufacturers
  - "Reproducibility" discussed as a new aspect of software trust (e.g. in security/privacy circles)
- Distro specific
  - Debian-policy packages should be reproducible
- Generic/all distro
  - Members from all distros
  - All distro package managers know about reproducible builds
  - Post list of funded reproducibility projects
- Upstream/elsewhere
  - Interest outside FLOSS projects (closed software, internal development)

    - Convince the industry that reproducibility is important
    - "Ecosystem janitor team": group to make sure upstream sources remain available & mirrored with checksum
    - Collaboration between industry and community much more significant
    - Google Chrome reproducible and user verifiable
    - Upstreams should be encouraged to issue a statement about reproducibility of their systems
- Miscellaneous
    - Get as many people supporting bootstrapping efforts as we have reproducible people now
    - Get reproducible builds explained in computer science courses
    - Have reproducible builds logo

Community/Documentation

- Community
    - Meetings
        - Regular 2-month IRC/voice meeting
        - More reproducible builds cross project in person meetings
        - Small 3-5 person hack sessions targeting specific goals
    - Network tooling/services
        - A community publishing logs about rebuilds exists
        - Central web service for submitting/retrieval of buildinfo files
    - Collaboration
        - notes.git will be more widely shared among distros
        - Have more reproducibility teams in other distros
        - Cross distro outreach
- Documentation
    - https://reproducible-builds.org website
        - Move content from Debian pages to https://reproducible-builds.org
        - Open archive of all reproducible builds talks that everyone can use
        - “Get involved” section
        - Common (cross distro) buildinfo format specification written

        - Blog planet/aggregator
        - Building a standard base for =.buildinfo= files
        - All cross project tools (not only project specific tools) documented
    - Projects or distributions
        - Fedora reproducible documentation on how to reproduce builds
        - Producing a template for projects that wish to make a public statement about reproducibility
        - Link to Nix docs from reproducible-builds.org
        - Improved reproducible builds-related documentation on the Bazel website
        - “How to attach signatures without disrupting reproduce?” guidelines for packagers.
    - Miscellaneous
        - Searchable database of reproducibility issues (e.g. manpages)
        - Examples of non-reproducibility problems explained (3-4 package examples — what did it cause?)
        - Including in the FAQ section tips on how to use various tools in a reproducible way
        - Encyclopedia for working around common non-reproducibility issues
        - Index collection with repro issues and solutions with search
        - Reproducible documentations as manpages
        - HOWTO about creating reproducible packages
        - Teaching people how to easily build and test existing software
        - Documenting setting up tests.reproducible-builds.org type infrastructure

Coverage/Upstream

- Common
    - Upstream projects adopting SOURCE_PREFIX_MAP
    - Reproducibility stats collected and reported (for Debian, FreeBSD, MacPorts, OpenWrt/LEDE, coreboot, NetBSD, F-Droid, Arch Linux, etc.)
    - Upstream git repos commits after cutoff date
    - Reproducible iso images
    - All distros publish a statement about reproducibility

- FreeBSD
  - Reproducible base system 100%
  - FreeBSD packages 90% reproducible by default (by count of pkgs)
  - All reproducible build options on by default
  - Expectation for reproducibility understood by FreeBSD committers
- Other Distros
  - MacPorts will be 50% reproducible by count of pkgs
  - More distros and FLOSS OSes joining reproducible builds
  - OpenWrt & LEDE to 90% reproducible builds
  - Proprietary software incorporating reproducibility
  - First 100% reproducible OS (with packages too)
  - Reproducible pkg src packages
- RPM-based
  - Support for buildinfo in necessary tools
  - Common discussion space for sharing problems and solutions
  - Some simple RPM pkgs already reproducible
  - Tools to reproducible builds
- Guix
  - GNU Guix will have measurements of pkg reproducibility
  - GNU Guix will have fixed repro issues in core packages: Guile, Python, GCC
- Debian
  - Debian unstable being 95% reproducible
  - Maintainers wanting to make reproducible debs
  - buildinfo in archive
  - Blocking Debian testing migration on reproducibility regressions
- Arch Linux
  - Writing documentation
  - Tools for users to verify reproducibility
  - 100% reproducible core repo
- NixOS

- Milestone to be reached: making NixOS minimal ISO (and all dependencies) reproducible
- Hydra (CI): graph or reproducibility progress
- Writing nixpkgs documentation
- Investigating reproducibility for OSX in nixpkgs
- Nix: allow users to configure desired trust level (i.e. "wont only want only binaries built by N out of K builders)
- Providing a verifiable bootstrap chain in nixpkgs going back ~10 years

Hardware/Embedded

- Hardware
  - How can we improve trust in hardware?
  - Open Hardware specs
  - Reproducible RISC-V implementation on FPGA
  - Where are reproducible/verifiable CPUs/microchips?
  - Reproducible HDL synthesis deemed feasible
  - Will have built/ported a free system to OpenRISC
  - How do we improve trust in our networking hardware?
- Firmware
  - Binary blobs are evil – how do we get more open source firmware?
  - Reproducible OS images
  - Tool to generate reproducible image
  - How can end-users verify their firmware?
  - Can we convince companies to open source firmware?
  - Can we have firmware audits with checksums by independent third parties?
  - Having more laptops supported without Management Engine
- Internet of Things
  - Having first devices with >= 50% reproducible software
  - Security updates for IoT devices will be a thing
  - Massive IoT botnets
  - Governments to force IoT manufacturers into maintenance!

    - Devices to be sold with clear maintenance period labels
    - Yocto/OpenEmbedded builds provide buildinfo and are reproducible
- Manufacturing
    - Convincing manufacturers to give out reproducible firmware
    - Starting conversations with vendors over benefits of reproducibility
    - Talking to OEMs/ODM/etc.

Dev Tools/Toolchain

- Metadata, etc.
    - PoC for opt to verify signature by multiple rebuilders
    - buildinfo. d. n to publish buildinfo of builds it _wants_ to have
- Compilers
    - GCC patches to upstream!
    - Rust reproducibility patches (inc. buildpath indep debug info)
    - Popular toolchains (C++, Java) work reproducibly and support SOURCE_PREFIX_MAP
    - LLVM/Clang on par with GCC regarding reproducible "patches"
        - Side note: Someone in the Clang community opened a review to add -ffixed-date-time for reproducible builds, not aware of SOURCE_DATE_EPOCH. A link to the spec was added in the review at https://reviews.llvm.org/D23934
    - Ocaml has no more reproducibility issues
    - Intentional nondeterminism patches
- Developer utilities
    - Git properties as "secure VCS" are analyzed
    - PDF generating docs reproducible!
    - R data files are reproducible
    - Strip-nondeterminism is smaller due to issues fixed "properly"
- diffoscope
    - Parallel diffoscope
    - More supported file formats
    - Integration with Taskotron
    - FreeBSD pkg format

    - Waterfall/time-series graph at what diffoscope is doing (like Chrome Dev tools -> Network tab)
    - Detection of order difference in many files
- Reprotest
    - Upstream devs are using reprotest to check for reproducibility
    - Reprotest has great usability
    - Reprotest runs on many different platforms
- Bootstrapping
    - Have fully mapped out bootstrapping chains for GCC, GHC, JDK, FPC, Gradle, Maven
    - Raising awareness about bootstrap binaries & shared work on GNU toolchain bootstrap
    - User accessible tools for fully bootstrap compilers
    - Compilers will be buildable with at least one other compiler.
    - Tool to cross-bootstrap any Debian arch
- buildinfo files
    - Defining a way to _select_ which buildinfo record should be compared
    - Automated buildinfo creation/inspection/comparison tool
    - Extra optional fields in buildinfo to help identify more causes of unreproducibility
    - Translation tool from buildinfo specs to Guix/Nix derivations + vice versa
- Specific toolchains
    - RPM toolchain(s) analyzed for sources of non-reproducibility
    - Reproducible autotools
    - Reproducible ELF Tool chains
    - No more buildpath issues (SOURCE_PREFIX_MAP widely adopted)

User tools

- Predictions
    - Make F-Droid.org provide simple links to reproducibility results
    - A tool to find who has been able to reproduce a package
    - Package managers allow installing reproducible packages only
    - FreeBSD pkg will policy/query for attributes like reproducibility or binary transparency

    - Users have the possibility to only install reproducible packages
    - Users able to gather statistics about reproducibility of packages installed on their system
    - Having a reproducible build "notary witness" server runnable by everyone
    - Tools available to give system reproducibility stats
    - F-Droid to have a binary transparency log for all releases
    - Script to rebuild a package and compare your results against the distro's
    - Guix will allow users to select "k of n" binary providers
    - Guix will have a "health" command listing CVEs and suspicious (non-reproducible) binaries
    - Tool for any user to execute reproduce instructions
    - Tooling to create "reproducibility transparency" logs exist
- Unknowns/questions
    - How to get fixes accepted by the Android Tools team at Google
    - Can a crowdsourced database of reproducibility test results limit its search space enough to be useful?
    - Will enough test infrastructure exist to establish reproducibility confidently?

Testing infrastructure

- Analyzing non-RB
    - Automatically classifying the causes of non-reproducibility
    - Advanced statistics (research on trends, tendencies, etc.)
    - Static Analysis in order to detect possible non-reproducible bugs in source code
- Miscellaneous improvements
    - Rebuild for each variation, to detect what effects the output
- Dev helping
    - Maintainers can upload packages to test for reproducibility
    - Creating a "fuzzing" tool to modify time|date|hostname|cwd etc. so that devs can test whether that affects them
- Deploy
    - Testing infra should be reproducible
    - Ability to setup tests.r-b.org locally to test patches

    - Ability to set up build for a pkg locally without whole CI infra (too)
- Non Debian
    - ci.freebsd.org will run the testing infra for packages
    - GNU Guix will be on https://tests.reproducible-builds.org
    - Measuring cross-arch reproducibility of noarch RPMs
- Debian tests
    - Testing arch:all separately
    - Testing arch:all cross-architecture
    - One build on one arch, per Jenkins.d.n job-run (not just in pairs build1+build2)
- Data storage
    - PostgreSQL is used
    - Storing all the build artifacts of completed builds

New directions

- Defining secure VCS (Version Control System)
- Understanding/analyzing security properties of Git
- Running transparency log for software
- Debian FTP archive distributing build info files
- Better cross distro build info support
- Designing security-related logic, workflows and algorithms for buildinfo files
- Having more people even at the Summit
- Crypto-signing more (reproducible) releases
- Sharing distributed databases of hashes for reproducible releases
- Having raised awareness of bootstrapping
- Start working on reproducible package installs
- Encourage projects, distros and software maintainers to publish a statement about reproducibility of their stuff
- Storing all build outputs...run diffs later to understand more
- Efforts on making major compilers bootstrappable has started
- Incorporating reproducible build results into binary transparency log
- Defining schema for records so that two different paths to build same artifact looks good

- Writing SOURCE_PREFIX_MAP specification
- Pushing GCC guild path patches
- Pushing SOURCE_PREFIX_MAP into other build tools
- Achieving 97%+ reproducibility with build-path independence
- Government to require reproducible builds by law for critical infrastructure software

Recommendations

Based on work done to date by the reproducible builds community, as well as the specific input received from community members who participated in Reproducible Builds Summit II, the event co-organizers believe that the following efforts would benefit from future support, resources and investment.

Hold a third Reproducible Builds Summit in 2017

The first two Summits showed how much value lies in providing the developer community with a recurring opportunity to connect, collaborate, and plan new initiatives.

Strategically designed in-person meetings can facilitate knowledge and skill sharing, create the space for collaborative efforts to start and grow, allow for productive and swift feedback loops.

The outcomes achieved by the reproducible builds community after the first two Summits demonstrate the importance of such convenings.

Improvement of documentation and developer training materials, both distribution-specific and cross distro

The remarkable amount of time and work dedicated to discussing, improving and developing new training and documentation materials during the second Reproducible Builds Summit highlights the need for dedicated resources to support such efforts.

Thorough and well-maintained documentation and training materials allow more developers to join the work on reproducible builds, enable current contributors to better advance their knowledge and skills, and empower the whole community to build and strengthen its capacity.

Development of cross distro resources

Cross distro research and development allows for increasingly shareable and interoperable processes across the range of projects working on reproducible builds.

Among the key efforts needing support are:

- Creation and sustainable maintenance of databases to track shared issues;
- Per-tool reproducibility issue tracking;
- Dedicated support to improve the management of upstream relationships;
- Comprehensive distribution-to-distribution package tracking.

Improved testing infrastructure

The https://tests.reproducible-builds.org test infrastructure allows for reproducible tests of any project. As discussed during the Summit, its progress is critical for the advancement of all the projects working on reproducible builds, and considerable work and resources are needed to support its operations.

Among the most immediate needs:

- Reproducing and documenting the test environment set up;
- Documenting notes useful for projects other than Debian;
- Merging the postgres code and create new schema;
- Making the database schema more cross distro friendly;
- Doing outreach and providing support to developers of projects other than Debian to do reproducible tests and share their outcomes.

Development of advocacy and policy-focused resources

The importance of the work on reproducible builds may seem obvious to the technology projects working in this field. However, the most relevant developments on reproducible builds are fairly new to many and it is therefore essential not to underestimate the importance of developing advocacy and policy-focused resources.

Clear and strategic use cases, designed to engage different personas, should be articulated and widely promoted in order to support the cause of reproducible builds across technical, legal and corporate sectors.

Increased participation of reproducible builds community members in sector specific conferences and convenings

The notable number of talks given by reproducible builds community members during the past year, and the very positive feedback received, attest for the importance of supporting developers in producing and giving presentations at relevant events.

The increasing presence of reproducible builds in conference programs helps to place the topic on the map of relevant efforts in play, and exposure to diverse developer communities provides opportunities to share knowledge with peers and collaborators-to-be.

The Reproducible Builds Summit organizers welcome discussion on these recommendations.

Appendix 1: Event agenda

Day 1 – Tuesday, December 13

10:00 Opening session

The co-organizers of the Reproducible Builds Summit II welcomed the group, and participants introduced themselves. The Summit facilitator shared an overview of the agenda, participation guidelines and meeting logistics.

11:00 Project showcase

Representatives of 9 project involved in the reproducible builds community hosted small-group Q&A conversations. The following is the list of the showcased projects:

- Reproducible FreeBSD
- Reproducible coreboot, OpenWrt, LEDE
- Building reproducible Tails ISO images
- RPM and reproducible builds in openSUSE
- F-Droid and reproducible Android apps
- Eliminating absolute build paths from debugging info
- Tests.reproducible-builds.org. How we constantly test Debian
- Buildinfo.debian.net. An experiment in distributing build information at scale
- Reproducible Debian status

12:15 Agenda brainstorming

Participants were invited to articulate which, in their opinion, were the most critical topics to be addressed during the Summit.

13:20 Lunch break

14:30 Working sessions I

A first set of working session was proposed to the group and small-group discussions were held. The session which took place during this time slot were:

- Diffoscope: Current status, challenges and next steps
- Reprotest: Project overview and potential improvements
- How to improve the reproducible builds documentation

- How to enable the user verification process
- Embedded systems and reproducible builds
- How to make RPM reproducible I

16:00 Closing plenary

The day ended with a closing plenary, during which participants were invited to share what topics they thought should be prioritized by the agenda of the following day.

16:30 Adjourn

Day 2 – Wednesday, December 14

10:00 Opening session

The session opening the second day of Summit aimed at highlighting the productive work done so far, and introducing the agenda for the day lying ahead.

10:20 Working sessions II

A new set of working sessions took place. The following are the discussions which took place:

- How do buildinfo files work
- How to make RPM reproducible II
- How to share the work done by different projects on reproducible images
- How to define a reproducible build I
- What end user policy should reproducible builds have?
- Improving the test infrastructure
- Fixing Gettext

12:30 Skill share sessions

Any participants who wanted to share a skill relevant to the Summit's work was invited to host a concise Q&A conversation with one or a few more interested peers. The following skill share sessions took place:

- Git-based packaging
- How to use C sanitizers and fuzzing
- How to make storage deduplicate and incentivize reproducible builds hash
- How to use buildinfos to analyze/test reproducibility
- Fedora ask-me-anything
- How to use Emacs
- How to run a start-up
- How to apply for funding from the Core Infrastructure Initiative
- How to improve cross distro packaging https://maintainer.zq1.de
- How (not) to use iframes on awesome webpages
- How to sign code in git (and correctly verify the signatures)
- Ask me anything about building on OSX

- How to do automatic hardware testing

13:00 Lunch break

14:20 Working sessions III

A third set of working sessions was proposed to the group. The sessions which were held were the following:

- What else for the auditable ecosystem?
- SOURCE_PREFIX_MAP
- How to improve the reproducible builds documentation II
- How to define a reproducible build II
- Writing a statement about what it means to do bootstrappable compilers I
- Reproducible builds use cases
- Using reproducible build information to help GPL compliance

16:00 Closing plenary

The second day closed with an invitation to share proposals for hacking projects to collaboratively work on.

16:30 Adjourn

Day 3 – Thursday, December 15

10:00 Opening Session

The third day of the Summit started with a brief opening session focusing on sharing the agenda overview for the final day of the meeting.

10:20 Working Sessions IV

The final set of working sessions has held. The following is the list of the discussions which took place.

- Cross Distro Collaboration on Reproducible Builds
- Writing a Statement About What It Means to Do Bootstrappable Compilers II
- How to Improve the Reproducible Builds Documentation II
- Using the Idea of “Certificate Transparency” for Packages
- State of Reproducible Builds

12:30 Project Hacking

13:15 Lunch Break

14:30 2017 Look Ahead Session

Participants were invited to visualize the goals worth pursuing in the coming year in different areas of the reproducible builds work. The areas they brainstormed on were the following.

- Advocacy/Outreach/Policy/Education
- Community/Documentation
- Coverage/Upstreams
- Hardware/Embedded
- Dev tools/Toolchain
- User Tools
- Testing Infrastructure
- New Directions

16:00 Closing Plenary

The Reproducible Builds Summit II closed celebrating the work done during the event, acknowledging the productive conduction of the meeting, and sharing appreciation for the knowledge and skills generously shared by all participating community members.

16:30 Adjourn