Open Technology Fund Community Lab: Reproducible Builds Summit

December 1-3, 2015. Athens, Greece

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

Table of Contents

Introduction

“We often speak as if open source software can't contain backdoors or malware because its source code is "published", rendering any potentially malicious code visible. But real-world software release processes have major transparency gaps that aren't addressed by most existing open source development practices. The biggest such gap is that compilation and packaging processes aren't reproducible.”

From: “Why and How of Reproducible Builds: Distrusting Our Own Infrastructure for Safer Software Releases” – Seth Schoen (Electronic Frontier Foundation) and Mike Perry (Tor Project)

Most aspects of software verification are done on source code, as that is what humans can reasonably read and understand. But most of the time, computers require software to first be converted into a machine-readable format in order to execute it.

Transforming the source code into binary format is called code “compiling” or “building”, and **reproducible builds** are achieved using **a set of software development practices which create a verifiable path from human readable source code to the binary code used by computers**.

When builds are reproducible, multiple parties can follow the documented build process independently and be confident they will achieve the same build result. We can thus gain confidence that a distributed binary image is indeed coming from a given source code base.

Why does this matter?

To provide a tangible example, we can reference to a recent popular attack.

In March 2015, The Intercept published from the Snowden leaks the abstract of a talk at an internal CIA conference in 2012 about Strawhorse: Attacking the MacOS and iOS Software Development Kit. The abstract clearly explains how unnamed researchers have been creating a modified version of Xcode that would — without any knowledge of the developer — watermark or insert spyware in the compiled applications.

A few months later, malware dubbed “XcodeGhost” was found targeting developers in order to make them unknowingly distribute malware embedded in iOS applications. Palo Alto Networks describes it as:

“XcodeGhost is the first compiler malware in OS X. Its malicious code is located in a Mach-O object file that was repackaged into some versions of Xcode installers. These malicious installers were then uploaded to Baidu’s cloud file sharing service for use by Chinese iOS/OS

*X developers."010816_*

A primary purpose of reproducible builds is exactly to resist such attacks. Recompiling these applications with a clean compiler would have made the problem easily visible, especially given the size of the added payload.

Several free software projects already, or will soon, provide reproducible builds. And this is truly promising since being able to trust the binaries enables others to perform a proper audit of the software, therefore **allowing us to determine that the software is uncompromised** and that it is not, for example, going to leak data, or contain trojans.

With the goal of supporting this emerging practice area, and to enable practitioners to build greater shared understanding and to cooperate on improving code and specifications, Open Technology Fund, Linux Foundation (Core Infrastructure Initiative), Google (Open Source Research) and Aspiration convened an experienced and diverse group of developers from more than 15 different free and open source software projects for the first-ever Reproducible Builds Summit, a 3-day meeting which took place in Athens, Greece, from December 1 to 3, 2015.

Summary

This report documents the work done by participants during the Reproducible Builds Summit and the outcomes achieved, and includes the following sections:

- **State of the Field** provides background and current status of the work on reproducible builds, along with areas where the community sees potential and room for improvement in the short and longer term;
- **Relevant Tools and Technologies** surveys the tools currently available to support production of reproducible builds;
- **Event Overview**, **Participating Projects, and Proceedings** document the event, from overall goals to participants to the content of the working sessions;
- **Outcomes, Next Steps**, **Recommendations** highlights emerging developments, noting post-event collaboration plans and providing recommendations for follow-up actions.

State of the Field

Background

The idea of reproducible builds is not new.

The topic has made occasional appearances in free software development discussion lists since 2000, and received a new wave of attention in 2008, with the development of the digital asset and payment system Bitcoin. Users of bitcoins needed a way to trust that they were not downloading corrupted software. Initial versions of Gitian, the software making the build process for Bitcoin Core executables reproducible, were written in 2011 in order to address this requirement.

The global surveillance disclosures of 2013 contributed to further raise interest in reproducible builds. Tor Project started working on making the Tor Browser build reproducibly in order to address the concern of “malware that attacks the software development and build processes themselves to distribute copies of itself to tens or even hundreds of millions of machines in a single, officially signed, instantaneous update” (see full article).

The success represented by making a piece of software as complex as a web browser build reproducibly demonstrated that it was feasible for other projects as well.

This prompted members of the Debian community to evaluate the opportunity to work on reproducible builds as part of their project. A first discussion hosted at DebConf13 was followed by the creation of a wiki page documenting the ongoing work on reproducible builds as well as a dedicated discussion list.

The wiki helped to disseminate more broadly the ongoing research and development efforts on reproducible builds, and to connect contributing developers, not only within the Debian community, but also across an ever-growing number of free software projects pursuing similar objectives.

In 2013 and 2014, further significant endeavors were pursued: several iterations on a file format to record the build environment, the creation of a generic tool that would post-process different file formats to remove timestamps or other source of non-determinism, an improved comparison tool, and the development of continuous testing for build reproducibility of the 22,000 Debian source packages.

2015 saw extensive work done to remove sources of non-determinism from the myriad tools involved in building packages.

The current state of uptake with reproducible builds

A remarkable number of **projects are working on reproducible builds.** Among them, and with the intent of presenting the following as a subset of an ever-growing list that the network aims to broaden, are: Arch Linux, Baserock, Bazel, Bitcoin, coreboot, Debian, F-Droid, Fedora, FreeBSD, Google, The Guardian Project, Guix, Homebew, MacPorts, NetBSD, NixOS, OpenWrt, Qubes, and Tor Browser.

Collaboration within the reproducible builds community takes place on the following **channels**:

- **Web site**
  - https://reproducible-builds.org
    The site provides information about the basic concept of reproducible builds as well as documentation, tools, updates about related events and an up-to-date list of participating projects.
- **Discussion lists**
  - https://lists.reproducible-builds.org/listinfo/rb-general
    The rb-general list is dedicated to general discussions regarding reproducible builds, aiming at improving collaboration between various software projects.
  - https://lists.reproducible-builds.org/listinfo/diffoscope
    The diffoscope list is a space to discuss the diffoscope tool (in-depth comparison of files, archives, and directories) amongst users and developers.
- **Internet Relay Chat (IRC)**
  - #reproducible-builds on irc.oftc.net
    The #reproducible-builds IRC is dedicated to real-time communication between community members.
- **Twitter**
  - https://twitter.com/ReproBuilds
    A social media account sharing updates about reproducible builds development, new documentation, in person events, write-ups.

As also discussed in a collective brainstorming session at the Reproducible Builds Summit in Athens, the community sees **great potential and room for improvement in a broad spectrum of fields where reproducible builds would be relevant.**

Open issues in reproducible builds

The following is a brief summary of current open questions, as aggregated by participants at the beginning of the Reproducible Builds Summit. This snapshot is not intended to be comprehensive, but it can be considered as representative regarding the state of the field.

The bullets represent actual text from post-it notes created in a participant brainstorm. Some items have been copy edited to improve readability.

Advocacy

- How do we make developers, users, companies, care about reproducibility?
- How can we make reproducibility a selling feature for free software?
- Can fast builds or build caching spur adoption of reproducible builds?
- Enumerate the use cases, customers and “business” reason to use/ adopt reproducible builds
- Take advantage of reproducible builds for build performance

Reproducible Builds Community

- Coordinate on changes to compilers and tools
- Define a standard representation of provenance (code, config and toolchain)
- Can we do anything cross-distro?
- How to stay connected/ share/ collaborate despite being on different projects?

Documentation

- Improve the documentation on reproducible-builds.org
- Build a shared database of issues with reproducible builds and patches

Challenges

- What are the biggest obstacles to reproducible builds other than time stamps?

Reproducible Builds Tools

- Building tools for upstreams to make their projects reproducible at development time
- Improve diffoscope
- Is there a right way of doing reproducible builds or do we need different tools?

Compilers

- How to handle builds that use profile-guided optimization?

- What efforts are needed to make Haskell/GHC reproducible?

Tool chain

- How to achieve identical binaries?
- Where to start with reproducible rpm build?
- Handling parallelism in compression tools in order to make the result deterministic
- Improve how expired signatures are handled

Build Environment

- What cross-platform technologies are there to guarantee a clean build environment?
- Should different build paths be supported?
- Figure out a common sandbox format/ abstraction to “build step”
- Can we share the tool chains to produce Windows and OS X binaries with Linux cross compilation?
- Discuss building on and for non-Linux platforms
- Compare methods for getting a “clean” build environment
- What's the fastest way to set up an environment for reproducible builds?
- How to enable diverse double distribution bootstrapping?
- Address the remaining barriers to actually doing diverse double-compilation (DDC)

User Aspects

- User verification and usability
- Ways of distributing the results of independent builds
- Find a good way for users to verify a release
- Integration for user tools indicating who has reproduced binaries

Testing

- How to test if a build is reproducible?

Certification

- Should we make a public database of reproduced binaries signatures?
- How to make a distributed code/package signing, for example to check N out of M signatures of a package to install it?
- Provide a way to prove an object was reproducibly built
- Multi signature distribution
- How to connect an artifact with an exact source snapshot, from which it was built?
- How can we trust binaries built in a P2P/decentralized way?

Funding

- How to get and distribute funding on reproducible builds
- Funding work on reproducible distros

System Bootstrap

- How to reproducibly install the system? aka reproducible bootstrap output
- How to build file system reproducible?
- Discuss system bootstrap: how to reduce the set of initial binaries needed to build a full GNU/Linux?
- How to build (reproducibly) file system images?
- Reproducible installs/ images

Relevant Tools and Technologies

Build reproducibility can be summarized in three essential requirements:

- The **build system** needs to be made entirely **deterministic**: transforming a given source must always create the same result. Typically, the current date and time must not be recorded and output always has to be written in the same order.

- The set of **tools** used to perform the build and more generally the build environment should either be **recorded or pre-defined**.

- **Users** should be given a way to **recreate** a close enough build environment, **perform** the build process, and **verify** that the output matches the original build.

Achieving reproducible builds for a given software project can require changes–usually small – to the project build system, along with a strategy on how to enable others to recreate an environment in which the builds can be reproduced.

To support this process, the **reproducible-builds.org** site provides **documentation** on how to achieve deterministic builds, how to define a build environment, how to distribute the environment, comparison protocols, and specifications.

Furthermore, different **tools** are currently available to support the work on reproducible builds:

- **diffoscope**
  diffoscope aims to investigate what makes files or directories different. It will recursively unpack archives of many kinds and transform various binary formats into more human readable forms for comparison. It can compare two tarballs, ISO images, or PDFs.

- **disorderfs**
  Problems with unstable order of inputs or other variations introduced by filesystems can sometimes be hard to track down. disorderfs is an overlay FUSE filesystem that introduces non-determinism into filesystem metadata. For example, it can randomize the order in which directory entries are read.

- **strip-nondeterminism**
  Some tools used in build systems might introduce non-determinism in ways difficult to fix at the source, which requires post-processing. strip-nondeterminism knows how to normalize various file formats such as gzipped files, ZIP archives, and Jar files. It is written in Perl with extensibility in mind.

Event Overview

The Reproducible Builds Summit convened at Impact HUB Athens, Greece from December 1 to 3, 2015.

Goals

Overall, the goal of the event was to better connect and coordinate members of this emerging community. Stated working goals of the meeting included:

- Updating and exchanging knowledge about the status of reproducible builds in various projects;
- Improving collaboration both between and inside projects;
- Helping to kick-start reproducible builds in other projects;
- Working together and hacking on solutions;
- Brainstorming designs on tools enabling end-users to get the most benefits from reproducible builds;
- Identifying existing and new infrastructure and collaboration that should be leveraged for the community.

Event program

The event was structured as three days of collaborative working sessions, along with optional evening activities.

The agenda was designed as a combination of planned sessions and participant-driven discussions, and specific topics were placed into time slots based on input at the meeting from those who were in attendance. Sessions were dialog- and outcome-oriented rather than presentations or lecture formats.

The program was designed to enable deeper collaborations and learning across the network of participants in order to collectively improve skills, strategies and impact in each other's respective efforts.

The event was documented in real time, and agenda and session notes were stored on an online etherpad shared with the participants, who were also invited to add notes and links during and after sessions.

The session notes have been published on the Reproducible Builds website at https://reproducible-builds.org/events/athens2015/agenda.

Projects Participating in the Event

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

Arch Linux
Arch Linux is a lightweight and flexible Linux distribution.

Baserock
Baserock is an open source project which integrates a large set of open source and free software components into complete custom operating systems. It combines elements of a Linux distort, build system, a workflow and a development environment all in one.

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

Debian
Debian is a free operating system, also coming with over 43000 packages and precompiled software bundled up for easy installation.

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

Fedora
Fedora is an operating system based on the Linux kernel.

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

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

Homebrew
Homebrew is a free and open-source software package management system that simplifies the installation of software on Apple's OS X operating system.

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

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

OpenWrt
OpenWrt is a Linux distribution for embedded devices.

Qubes
Qubes is a security-oriented, open-source operating system for personal computers.

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

Proceedings

Facilitated and co-organized by Allen Gunn and Beatrice Martini of Aspiration, in collaboration with Jérémy Bobbio (Lunar) and Holger Levson from the Debian project, the program alternated between rounds of swift collective brainstorming and focused small-group discussions.

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/athens2015/agenda.

Day 1 – Tuesday, December 1

The convening was opened by words of **welcome** from the hosts, brief participant **introductions**, along with agenda, participation guidelines and logistics **overviews**.

Participants were then invited to join the first session of the day: an **interactive Q&A** focusing on "Reproducible builds in Debian – where are we at today?"

Five Debian community members – Jérémy Bobbio (Lunar), Reiner Herrmann, Mattia Rizzolo, Holger Levsen, Chris Lamb – facilitated conversations on the topic, addressing questions at different levels of experience, from introductory to advanced. Participants were asked to self-assess their knowledge of the topic by responding to the statement "I understand reproducible builds extremely well", positioning themselves along an imaginary line going from 0 to 100%. Five small groups were formed accordingly.

The session provided the opportunity for Debian community members to share information about the work done on reproducible builds to date, while participants representing other projects had the chance to ask questions, share feedback and compare experiences, as well as discuss challenges and opportunities for further cross-network collaboration.

After a quick break, participants joined a **speed geeking** session, a fast-paced knowledge sharing format that enabled participants to quickly engage in a number of presentations on a range of related projects. Five facilitators were invited to host stations to present their work or project; participants were split into five groups to visit the stations in 5-minute rounds.

The following topics and projects were presented at the stations.

- **Binary Transparency, Daniel Kahn Gillmor**
  Binary transparency is a key practice to protect users from illegal surveillance or other corruption of binaries. Providing public access to source code and adopting a

reproducible build process give the opportunity to verify the integrity of pre-compiled binaries. Software should be distributed with verifiable signatures from a trusted party and a path for users to verify that their copy of the software is functionally identical to every other copy (a property known as "binary transparency").

- **Tor Browser, Georg Koppen**
  Tor Browser is a privacy-enhanced web browser using the Tor network and running on Windows, Mac OS X, and Linux operating systems. The Tor Browser team uses Gitian to ensure that byte-for-byte reproducible packages can be built from the source repository by anyone.

- **Bazel, Laurent & Łukasz Zemczak**
  Bazel is Google's own build tool, now publicly available in Beta. Bazel has built-in support for building both client and server software, including client applications for both Android and iOS platforms. Bazel strives for quick and correct builds, and both aspects require fully deterministic build processes.

- **Guix/ Nix, Ludovic Courtès & Eelco Dolstra**
  Nix and Guix are both “purely functional” package managers enabling very accurate dependency tracking and fully isolated builds. NixOS and GuixSD are full operating systems based on Nix and Guix respectively. Entire systems can be described programmatically and thus easily replicated, with bit-for-bit reproduction as future goal.

- **F-Droid, Hans-Christoph Steiner**
  F-Droid is an installable catalog of free and open source software applications for the Android platform. The client makes it easy to browse, install, and keep track of updates on the device.

The morning ended with an **agenda brainstorming** activity, during which participants were invited to answer the question “What would you like to work on during the Summit?” by filling out post-its notes, with either a topic statement or a question to be addressed. The notes were then arranged on the wall and clustered by theme, producing a topic map to inform and drive the remainder of the Summit agenda.

The afternoon was dedicated to **collaborative breakout sessions**. The sessions covered topics identified as high-priority based both on input received before the event as well as during the agenda brainstorming exercise.

Each session had one or two facilitators and a volunteer note taker identified among the participants. Participants who were not occupied with facilitation or documentation had the opportunity to join two different sessions in sequence, as the organizers allowed for two contiguous session slots to take place.

The following is a list of the sessions which took place, in two rounds, including a brief summary of their scope.

Collaborative breakout sessions I

Shared problems we can work on this week, Daniel Kahn Gillmor
Participants identified the following action items for exploration at the Summit: discussing what needs to go in a common buildinfo file; making a to-do list for reproducible builds documentation; identifying areas for improvement in related tools, especially diffoscope.

How to control the build environment, Jérémy Bobbio (Lunar)
This session surveyed a range of possible approaches to controlling build environments. Some projects set the same build environment for every developer; other projects want different environments to produce the same build results. The discussion focused on specific details relative to different scenarios (in Debian, FreeBSD, Qubes, Google, OS X) and identified potential action items, such as improving libarchive and archive tools; improving FreeBSD jails; improving Linux containers; and building a tool to run the same build process in different environments and compare results.

Generalizing reproducible.debian.net continuous test system, Mattia Rizzolo
A number of possible next steps were discussed, including generalizing scheduling, web page and common core; sharing notes about packages between different distributions; getting developers to tag packages in other distros to compare notes on different packages; and listing state of reproducibility of a package in different distributions.

Distribution/compiler bootstrapping and whole chain auditing, Ludovic Courtès
Topics covered included toolchain boostrap status (in Nix, Guix, Debian), Diverse Double Compilation, wishes and possible next steps for compilers including certification for compilers; having a compiler written in C which can then be verified; create a compiler ring.

Collaborative breakout sessions II

Advocacy: communicating the benefits of reproducible builds, Holger Levsen
Target stakeholders for advocacy efforts were identified, specifically software developers, distro developers, end users, and managers. Myths identified to be addressed included “It is hard to check whether software is reproducible”; “Timestamps, username, hostname in documentation are useful”; “Reproducible builds are hard and/or impossible”. Rebuttals provided to address these myths include the facts that audit trails are no longer needed with reproducible builds, and that over 19K packages are already reproducible.

Documentation road map, Jérémy Bobbio (Lunar)
Among the resources considered for creation and addition to the existing documentation: best practices for an upstream; checklists for developers; distro-specific documentation; index by tool showing information, related issues, and how to resolve them; documentation for non-free

software tool chains and platforms that might not be able to achieve the bit-by-bit paradigm.

Making file system images reproducible, Marek Marczykowski-Górecki
Two problem areas were discussed: building installation media reproducibly, and building base images reproducibly. Conclusions reached included: developers need to check and ensure that the command line tools for image generation are deterministic, and diff support for filesystem images would be helpful, possibly by adding that support to diffoscope.

GHC, Joachim Breitner
The discussion spanned a range of topics on GHC, including the possibility of writing a tool to renumber random numbers in object files to enable parallelization; rebuilds on Debian caused by Haskell ABI changes; triggers for rebuilds, the associated waste of CPU time, and how to avoid that; and ways to prevent gcc from including the path to temp files.

The first day of the event ended with a **closing circle**, during which participants were invited to weigh in on what had been most useful during the course of the day, and share their goals and priorities for the agenda of the second day by answering the question “What is one thing you would like to work on or get done tomorrow?”. The organizing team took note of the requests in order to tailor the Day 2 agenda accordingly.

Day 2 – Wednesday, December 2

The second day of the convening started with **opening circle**, summarizing the work done during the first day and offering an agenda overview for the day ahead.

The first half of the morning was dedicated to **collaborative breakout sessions**.

Collaborative breakout sessions III

RPM, Dhiru Kholia & Holger Levsen
Discussion about development of the RPM package manager focused on automatic rebuilding tool for users; buildinfo files in rpm; SOURCE_DATE_EPOCH support in rpm; custom rpm repository with patched packages to be be setup on people.fedora.org; and packaging diffoscope in Fedora.

Diffoscope wish list, Jérémy Bobbio (Lunar)
diffoscope aims to get to the bottom of what makes files or directories different. Among the diffoscope wish list items discussed: writing a wrapper around diff that strips out large identical chunks and only feeds in smaller chunks so that diffing can work on large files; multithreaded/parallel processing; better/smarter ELF diff'ing; sandboxed environment to better secure diffoscope access of untrusted input; have a hosted web service for doing the diffing (will require a sandbox); internal anchor links in the HTML report to be able to jump around; have an HTML server so that you can view diffing results coming in in real-time.

Shared database of issues of non-determinism, Daniel Kahn Gillmor
Database aspects discussed included use cases for sharing notes; issues, meta-issues, and their characteristics; history (linking to example patches and keeping old issues); linking packages between distros; notes maintenance (need of a format where it's possible to separate "upstream" issues, distro-specific issues, distro-specific versions, etc; considering that different distros might have different way to fix issues); notes format.

Use cases and success stories for build reproducibility, Daniel
Participants discussed a series of benefits that reproducible builds provide in different context (such as Debian, Google, Tor, F-Droid, NixOS, MacPorts). Categories of the benefits were identified: financial benefits, security benefits, infrastructure benefits, reduction of responsibility on single developers, reduction of external trust. Planning to add documentation of use cases on the reproducible-builds.org site would likely require a categorization by target audience, i.e. benefits for users, for developers, for companies.

During the second half of the morning participants got together to take part in a **skill share exercise**. In this format, participants were encouraged to share any skill they considered relevant to the meeting scope during a 30-minute slot. The session was structured so as to minimize group size and maximize 1-on-1 sharing opportunities.

A survey **of the skill share conversations** which took place included:

- How to get reproducible buildroot for Fedora / What needs to be done to get Arch Linux reproducible
- How to rebuild .debs from reproducible.debian.net
- How to do git [tag] signatures correctly (and verify them)
- How to chroot and mount on Linux without superuser privileges
- How to prove Gödel's First Incompleteness Theorem
- How to mass bug file in Debian
- How to automate Jenkins configuration and set up (in LI plugins)
- How to edit ZIP archives
- How to build reproducibly in OS X
- How to replace binutils with an alternative toolchain
- How to fix timestamps issues to make builds reproducible
- How to set up a system to have "everything" go through Tor

The afternoon was dedicated to **collaborative breakout sessions** and **hacking**.

Collaborative breakout sessions IV

Becoming a diffoscope hacker, Jérémy Bobbio (Lunar)
Topics covered during the session: fundamental information about diffoscope (how it's written, where to find the testsuite); filetype support; files; how to implement comparison; how to run commands on files; how to support archives and containers; how to submit patches.

Communicating reproducible builds to users, Niels
The session unfolded around three key questions: What is the mechanism for publishing and sharing the results of reproducible builds? How do end users verify a reproducible build? What happens when verifiers can't agree on the result of a reproducible build? Possible solutions and further open questions were highlighted and noted for future discussion.

Post-event collaboration, Georg Koppen
Participants discussed benefits from collaboration; currently existing infrastructure (web server, git repo, jenkins setup, mailing lists, Twitter account); issues and challenges presented by collaboration; opportunities for further periodical in person meetings.

.buildinfo file, Daniel Kahn Gillmor
Key questions addressed by the discussion: What should we have in the .buildinfo file? What currently affects the build but isn't documented? What explicitly shouldn't affect the build and should we record that?

SOURCE_DATE_EPOCH, Ximin Luo
SOURCE_DATE_EPOCH is a distribution-agnostic standard to provide a pre-defined timestamp to build systems, based on the number of seconds since the epoch, excluding leap seconds. On the reproducible-builds.org site there are specs for SOURCE_DATE_EPOCH, the current upstream packages which have been fixed, and patch examples. One takeaway: distros need to get build tools to read this environment variable, and the tools should use this instead of the current date and time.

Hacking Time

After report backs on the breakout sessions were completed, the program shifted to "**hacking time**", where participants engaged in hands-on collaborative coding tasks to improve tools and make specific code bases more reproducible. Participants shared their respective hacking plans, and working groups formed from there.

The second day of the event ended with a **closing circle**, during which participants summarized key outcomes from the hacking time slot, and were invited to share their answer to the question "What is one thing you would like to work on or get done tomorrow?". The organizing team took note of the requests in order to tailor the agenda accordingly for the third day.

Day 3 - Thursday, December 3

The third day of the convening started with another **opening circle**, including a summary of the work done during the first two days and agenda overview for the day ahead.

The first half of the morning was dedicated to **collaborative breakout sessions**.

The following are the sessions which took place, including a brief summary of their scope.

Collaborative breakout sessions V

Advocacy next steps, Jérémy Bobbio (Lunar)
How to share knowledge and raise awareness on the relevance of reproducible builds? Among the next steps discussed: outreach to applications; integration into buildsystems; engage with key communities, institutions, companies and influencers who could also become strategic allies; give presentations at key events; consider working on press releases and media outreach.

Unsolved problems, Chris Lamb
Among the challenges discussed: time services (like snapshots.debian.org); infrastructure to host .buildinfo; durable service hosting; adding reproducibility aspects to GNU/Apache/etc software policies; writing a "random environment" tool; certificates expiring in test suites, etc; build paths; core dumps from VM-heavy software (e.g. emacs, Smalltalk, etc); profile-guided optimization.

Next Reproducible Builds event, Holger Levsen
This session addressed the following aspects of doing a second Reproducible Builds Summit: desired outcomes of potential future in-person meetings; ideal frequency of these events; increased focus on emerging leadership to organize future convenings; ideal numbers of participants and strategic invitations and outreach; possible locations (criteria to be considered include local hacker community/support; local logistics allies; visas; costs; easy-to-reach location; accessible, welcoming and value-aligned venue).

Buildinfo file version 2, Daniel Kahn Gillmor
Participants mapped out build inputs ranked by importance for reproducibility, from “absolutely necessary” (human intent) to “too specific” (happenstance). The discussion helped identify five key purposes of build info: document what the package builder did; reproduce build environment; input to system to find minimum set required for reproducibility; for debugging non-reproducibility; forensics.

Hacking Time

After wrapping up the morning sessions, the second half of the morning was dedicated to more **hacking** on related projects.

The afternoon started with the final slot of **collaborative breakout sessions**.

Collaborative breakout sessions VI

Fundraising, Holger Levsen

The session focused on brainstorming on the following questions: Where could we get funding from for expanded Reproducible Builds efforts? (e.g. foundations, companies, individual donors)? How can the money be strategically spent? (e.g. build infrastructure; resources and developers' work; meetings and travel; hardware; advocacy) What are currently open questions? (e.g. how to allocate funds? How to be transparent and accountable?)

Describing the ideal reproducible build testing tool, Jérémy Bobbio (Lunar)
The discussion was structured as an exercise in README driven development, asking participants to describe the perfect tool to answer the question “How can I know if a piece of software is reproducible?”. The name given to such tool was “reprotest”, with the intent to target developers who want to test whether their software is reproducible.

Debian build info file, Daniel Kahn Gillmor
The final session on buildinfo files focused on various proposals for future developments, challenges to work through in that regard, possible next steps, currently open questions, and changes required to the buildinfo page on the Reproducible Builds wiki.

Signatures challenges, Wojtek
The session began with a review of why source code signing matters: it provides accountability and makes code review meaningful (clarifying exactly what code was audited). From there, participants discussed current challenges in signing technology and processes; who actually signs; documentation; key storage; threat models; binaries transparency log; status of verification of different paths; and what next.

The 3-day Summit drove to conclusion with a **Where From Here** exercise.

Participants were invited to reflect on all aspects of the event they just experienced, and then think about next steps, both for themselves and the group. They were then asked to write statements on post-it notes expressing their reflections, starting with one of three phrases:

- "I Will", indicating an action item an individual participant was committing to follow up on after the event;
- "We should", indicating areas where participants would like to see ongoing collaboration and follow up, and
- "Don't Forget", where participants were able to note unfinished business and pending matters for future endeavors.

Results were collected by the organizers and transcribed, and shared with participants in order to facilitate further collaborations and partnerships after the meeting.

The final day of the event ended with a **closing circle**, during which appreciations were shared and participants summarized key learning and outcomes from the Summit.

Event Outcomes

The Reproducible Builds Summit generated a range of outcomes, ranging from a strengthened community of practice to action plans to code committed to various projects and distributions, including the following:

- **General outcomes**
  - Broader base of knowledge and shared practice established **between 14 participating projects**
  - A number of projects left with **concrete plans** (including Qubes https://groups.google.com/forum/#!msg/qubes-devel/gq-wb9wTQV8/mdliS4P2BQAJ, and openSUSE https://hackweek.suse.com/13/projects/131)
  - **Deeper understanding of where and when reproducible builds are useful**
- **New and improved tools:**
  - **findnewest:** recursively find newest file in a hierarchy and print its timestamp useful to determine a good value for SOURCE_DATE_EPOCH https://github.com/0-wiz-0/findnewest
  - **diffoscope:** new contributors, many improvements and portability fixes
    Most changes were released in versions 43 and 44:
    https://tracker.debian.org/news/733139
    https://tracker.debian.org/news/733894
    Now available on Homebrew: http://brewformulas.org/Diffoscope
    MacPorts: https://trac.macports.org/browser/trunk/dports/sysutils/diffoscope
    and NetBSD pkgsrc: http://pkgsrc.se/sysutils/py-diffoscope
    Work was done on a FreeBSD port.
  - **Initial draft of specification for reprotest**
- **Patches submitted:**
  - Support for SOURCE_DATE_EPOCH in RPM:
    https://github.com/boklm/rpm/commit/5f09ea8feae8462c5ac20169694d1b52fea2c8d9
  - Posted patch for FreeBSD kernel reproducibility: https://reviews.freebsd.org/D4347 (Ed Maste)

- Support SOURCE_DATE_EPOCH in FreeBSD package manager: https://github.com/freebsd/pkg/commit/69ee4f6417f5fa5f8204e113deb82af3e73bb97f
- Export SOURCE_DATE_EPOCH in FreeBSD ports tree: https://reviews.freebsd.org/D4385
- Homebrew binary package reproducibility: https://github.com/Homebrew/homebrew/pull/46587
- Improvements on continous testing infrastructure hosted at reproducible.debian.net
- koji packaged for Debian to easier test RPM on reproducible.debian.net: https://anonscm.debian.org/cgit/reproducible/koji.git
- Updated documentation of state-of-art for RPM: https://github.com/kholia/ReproducibleBuilds
- Make grub-mkimage output reproducible: https://lists.gnu.org/archive/html/grub-devel/2015-12/msg00013.html
- Automatic rebuild and check for identical contents on GNU Guix: http://git.savannah.gnu.org/cgit/guix.git/commit/?id=07e70f4846521c1fa5319b25f23eea171a03fccd
- Support for fixed timestamp in mksquashfs http://sourceforge.net/p/squashfs/mailman/message/34673610/

- **Post-Event Blog Posts**
  - Joachim Breitner, Debian, “Reproducible Builds World Summit” https://www.joachim-breitner.de/blog/688-Reproducible_Builds_World_Summit
  - Georg Koppen, Tor Project, “Tor at the Reproducible Builds Workshop in Athens 2015” https://blog.torproject.org/blog/tor-reproducible-builds-workshop-athens-2015
  - Hans-Christoph Steiner, The Guardian Project: “First Reproducible Builds Summit” https://guardianproject.info/2015/12/09/first-reproducible-builds-summit
  - Clemens Lang, MacPorts, “Build Reproducibility Workshop Report” https://lists.macosforge.org/pipermail/macports-dev/2015-December/032016.html
  - Dhiru Kholia, Fedora, “Reproducible Builds for Fedora (a reboot)”

https://lists.fedoraproject.org/archives/list/devel%40lists.fedoraproject.org/message/DECB4Z2ZSLHPX4W3QPDEPHGWF6PJEII3

- Ludovic Courtès, GNU, “Reproducible GNU!” https://lists.gnu.org/archive/html/gnu-system-discuss/2015-12/msg00000.html
- Ludovic Courtès, Guix, “Reproducible Build Summit” https://lists.gnu.org/archive/html/guix-devel/2015-12/msg00107.html

Next Steps

Participants identified a rich array of recommended next steps during the final session. These recommendations are drawn both from sessions notes as well as from the final brainstorm.

Inter-project collaboration and shared resources

- Revive distromatch, a database to map package names from one distro to another. This would greatly help exchange of status and patches:
  http://www.enricozini.org/2011/debian/distromatch/
  http://www.enricozini.org/2011/debian/distromatch-deploy/
- Develop a cross-project database of issues affecting build tools and software
- Turn current test setup into tests.reproducible-builds.org and test more distros
- Interconnect these test results
- Set up source archival service for distributions which don't have their own (like history snapshot.debian.org)
- Set another cluster to do continuous reproducibility testing like reproducible.debian.net
- Develop reprotest
- Organize another meeting (and don't forget to invite folks from OpenBSD, DragonflyBSD, openSUSE, more Fedora/RPM, Gentoo, CentOS, Scientific Linux, Docker, Microsoft, Apple, Facebook, Oracle, Intel)

Documentation

- Improve documentation on reproducible-builds.org by adding a list of problems and solutions. Index by tools (e.g. gcc, clang, ghc, epydoc, tar, zip, etc.)

Pending issues to research and resolve

- Research how to cope with build path variations (gcc has two patches waiting already https://gcc.gnu.org/bugzilla/show_bug.cgi?id=47047 and https://gcc.gnu.org/bugzilla/show_bug.cgi?id=68848) but we know way more tools are going to write the full source path if we turn the variation back on
- Research how to handle software built with profile guided optimizations
- Research software who do core dumps as part of their build process (emacs, smalltalk)
- Investigate cross-building (e.g. building Debian on FreeBSD and vice-versa)
- Work on having a “binary transparency log” running for reproducible builds
- Add support for SOURCE_DATE_EPOCH to more tools (LLVM, etc.)

Advocacy about Reproducible Builds

- Reach out to big free software advocacy organizations – e.g. Open Source Initiative,

Free Software Foundation – to have them advertise reproducible builds as part of a good development practice.

- Help representative attend software development conferences
- Develop a small curriculum for computer science students
- Create leaflets to hand out at conferences

Recommendations

Building on the next steps outlined above, the event organizers make the following recommendations for future support and investment.

- **Convene a second Reproducible Builds Summit in 2016**, and consider the viability of recurring events past that point. While much was accomplished in Athens, all those present agreed that much work remains on the road ahead, and sustaining and growing this nascent community of practice during this critical phase should be the highest priority.

- **Invest in better developer training and documentation materials**, building on the resources already available at https://reproducible-builds.org, including case studies, success stories, threat models and other second-generation resources to drive broader and faster adoption of reproducible build practices.

- **Invest in development of advocacy materials** which articulate the benefits and make the case to senior managers, project principals and other influencers regarding the importance of reproducible builds in the FOSS ecosystem. While the mandate and benefit are clear to those already working in this area of practice, there are a number of stakeholders across free and open source communities who remain skeptical or unconvinced of benefits relative to cost or current practices.

- **Invest in better cross-project resources**, including databases to track shared issues, per-tool reproducibility issue tracking, support for better management of upstream relationships, and distribution-to-distribution package tracking.

- In conjunction with enhanced advocacy, **invest in increased presence at conferences and other convenings** where developers can learn about build reproducibility and move toward making their own code bases reproducible.

The Reproducible Builds Summit organizers welcome discussion on these recommendations.

Appendix 1: Event Agenda

Day 1 - Tuesday, December 1, 2015

Opening circle
Organizers welcomed the group, participants introduced themselves, and the program began with an overview of the 3-day event, including goals, format, and participant guidelines.

Interactive Q&A: Reproducible builds in Debian – where are we at today?
Debian community members shared updates on the state of the work on reproducible builds

SpeedGeeking
In a fast-paced interactive format, event participants working on related efforts shared their experiences and explained current work on reproducibility.

Agenda brainstorming
Collective brainstorming was done by the group to map the universe of topics and open questions to be discussed at the meeting and in ongoing work.

Collaborative breakout sessions I
Small-group working sessions focused on specific topics proposed and facilitated by participants.

Collaborative breakout sessions II
Working sessions continued.

Closing circle
The final plenary recapped of the work done to that point and invited participants to share plans and goals for the day ahead

Day 2 - Wednesday, December 2, 2015

Opening circle
The program began with a Day 1 summary along with a Day 2 overview.

Collaborative breakout sessions III
Working sessions continued.

Participant skill share
Participants share skills relevant to the meeting scope in a 30-minute slot

Collaborative breakout sessions IV

Working sessions continued.

Hacking time
Self-organized hacking focused to address tasks emerged during the meeting

Closing circle
Recap of the work done on Day 2 along with an invitation to share plans and goals for the third day.

Day 3 - Thursday, December 3, 2015

Opening circle
The program began with a Day 2 summary along with a Day 3 overview.

Collaborative breakout sessions V
Working sessions continued.

Hacking time
Self-organized hacking focused to address tasks emerged during the meeting

Collaborative breakout sessions VI
Working sessions continued.

'Where from here' exercise
Participants shared their reflections on the Summit, capturing action items and recommended next steps.

Closing circle
The event closed with final reflections, appreciations, and announcements about next steps.