Reproducible Builds

An independently-verifiable path from source code to software

Frédéric Pierret

Oct. 7th 2022
Frédéric Pierret (fepitre)

- PhD in Applied Mathematics,
- Modeling of dynamical systems,
- Designing build systems.

- github.com/fepitre
- frederic@invisiblethingslab.com
- frederic.pierret@qubes-os.org
What?
What are reproducible builds?

**Reproducible Builds**

enable anyone to reproduce identical binary packages from a given source
What?

source \[\rightarrow \text{build} \rightarrow\] binary
What?

free software

source → build → binary

freedom to study

freedom to run
What?

source → build → binary

can be verified → can be used
source → build → binary

can be verified

prove it to me!

can be used
Why?
Why?

Reproducible builds allow for independent verifications that a binary matches what the source intended to produce.

... and other nice things.
But I’m the developer!

“I know what’s in the binary because I compiled it myself!”

“I’m an upstanding, careful, and responsible individual!”

“Why should I have to worry about hypothetical risks about the contents of my binaries?”
But the build machines are secure

- How can you be sure?
But the distribution packagers released it!

- What’s in between build artifacts and signed-build artifacts?
Unpleasant thoughts

- We think of software development as a fundamentally benign activity,
  - “I’m not that interesting.”
- Users can be targeted through developers,
- Known successful attacks against infrastructure used by Linux (2003), FreeBSD (2013), PHP (2021) and some undisclosed.
Seriously... During a CIA conference in 2012¹:

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

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

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

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

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

¹https://firstlook.org/theintercept/2015/03/10/ispy-cia-campaign-steal-apples-secrets/
And yes...

- SolarWinds\(^2\) was exactly what would be **prevented by reproducible builds**!
- The compromission was on build servers!

\(^2\)https://www.sans.org/blog/what-you-need-to-know-about-the-solarwinds-supply-chain-attack/
How small can a backdoor be?

OpenSSH 3.0.2 (CVE-2002-0083) – exploitable security bug (privilege escalation: user can get root)

{  
    Channel *c;
    - if (id < 0 || id > channels_alloc) {
    + if (id < 0 || id >= channels_alloc) {
        log("channel_lookup: %d: bad id", id);
        return;
    }
}
Result of fixing the bug (asm)

cmpl $0x0,0x8(%ebp)       cmp $0x0,0x8(%ebp)
js 16                      js 16
mov 0x4,%eax              mov 0x4,%eax
cmp %eax,0x8(%ebp)        cmp %eax,0x8(%ebp)
jle 30                    jle 30
mov 0x8(%ebp),%eax        mov 0x8(%ebp),%eax
mov %eax,0x4(%esp)        mov %eax,0x4(%esp)
movl $0x4c,(%esp)         movl $0x4c,(%esp)
call 25                  call 25
Result of fixing the bug (hex)

<table>
<thead>
<tr>
<th>Vulnerable</th>
<th>Fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>55 89 e5 83 ec</td>
<td>55 89 e5 83 ec</td>
</tr>
<tr>
<td>28 83 7d 08 00</td>
<td>28 83 7d 08 00</td>
</tr>
<tr>
<td>78 0a a1 04 00</td>
<td>78 0a a1 04 00</td>
</tr>
<tr>
<td>00 00 39 45 08</td>
<td>00 00 39 45 08</td>
</tr>
<tr>
<td>7e 1a 8b 45 08</td>
<td>7c 1a 8b 45 08</td>
</tr>
<tr>
<td>89 44 24 04 c7</td>
<td>89 44 24 04 c7</td>
</tr>
<tr>
<td>04 24 4c 00 00</td>
<td>04 24 4c 00 00</td>
</tr>
<tr>
<td>00 e8 fc ff ff</td>
<td>00 e8 fc ff ff</td>
</tr>
<tr>
<td>ff b8 00 00 00</td>
<td>ff b8 00 00 00</td>
</tr>
<tr>
<td>00 eb 35</td>
<td>00 eb 35</td>
</tr>
</tbody>
</table>
Resulting difference in the binary

What’s the difference between \texttt{if} (a > b) and \texttt{if} (a >= b) in x86 assembly?

\begin{tabular}{l l}
assembly & JLE & JL \\
opcode & 0x7E & 0x7C \\
binary & 01111110 & 01111100 \\
\end{tabular}

A single bit!

Other corresponding opcode pairs also differ by just a single bit (JGE=0x7D, JG=0x7F)
Do not blame developers (all the times)

- Malicious modifications to binaries could result in irrevocable unwanted actions,
- Individual developers could be blamed for such modifications,
- Reproducible builds therefore *protect* developers
Nothing new though

From: Martin Uecker <muecker@gmx.de>
Cc: debian-devel@lists.debian.org
Date: Sun, 23 Sep 2007 23:32:59 +0200

I think it would be really cool if the Debian policy required that packages could be rebuild bit-identical from source. At the moment, it is impossible to independly verify the integricty of binary packages.

Wouldn’t it be cool?

- Debian is the largest collection of free software
- More than 25,000 source packages
- “Our priorities are our users and free software”
How?
How?

- Record the build environment
- Reproduce the build environment
- Eliminate unneeded variations
How:
Record the build environment
New control file *.buildinfo which records:

- Versions of build dependencies
  - ... and their dependencies
- Checksum of the source package.
- Checksums of the binary packages.
Example *.buildinfo

Format: 1.0
Source: apt
Binary: apt apt-dbgsym apt-utils apt-utils-dbgsym libapt-pkg-dev libapt-pkg6.0 libapt-pkg6.0-dbgsym
Architecture: amd64
Version: 2.2.4
Build-Origin: Debian
Build-Architecture: amd64
Build-Date: Thu, 10 Jun 2021 09:12:36 +0000
Build-Path: /build/apt-oBIw5E/apt-2.2.4
Installed-Build-Depends:
  autoconf (= 2.69-14),
  automake (= 1:1.16.3-2),
  autopoint (= 0.21-4),
  autotools-dev (= 20180224.1+nmu1),
  base-files (= 11.1),
  base-passwd (= 3.5.50),
...

28 of 73
How:

Eliminate unneeded variations
Eliminate unneeded variations

Make the build process deterministic:

Same input

=  

Same output
Dealing with variations: two approaches

- If a build differs because of X, you have two ways of dealing with it:
  - either make X always the same,
  - or make the build independent of X.
- Good example is build path:
  - Debian tries to not embed it,
  - Fedora or OpenSUSE always builds with the same path (e.g. Mock).
Investigating packages\(^3\)

- **diffoscope**
  - figure out what makes files or directories different.
  - 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 just as easily.

- **reprotest**
  - builds the same source code in *different environments*,
  - checks the binaries produced by the builds,
  - see if changing the environment, without changing the source code, changed the generated binaries.

\(^3\) [https://reproducible-builds.org/tools/](https://reproducible-builds.org/tools/)
How:

Reproduce the build
Reproduce the build for Debian aka **rebuild**

- **snapshot.notset.fr**
  - A **working** replacement of snapshot.debian.org,
  - History of Debian repositories from 2017 to now,
  - amd64, all and sources (upcoming arm64)

- **debrebuild**
  - Find the right archive snapshot,
  - Install packages listed in the *.buildinfo file,
  - Do the rebuild.

---

4 https://github.com/fepitre/debian-snapshot
5 https://github.com/fepitre/debrebuild
PackageRebuilder

Welcome to the PackageRebuilder instance hosted by Frédéric Pierret.

This page shows the results of verification builds of official distribution packages in the repositories in an effort to be fully reproducible. For more information read the Reproducible Builds website and the announcement Reproducible builds for Debian: a big step forward.

The source code is licensed MIT and available here.
beta.tests.reproducible-builds.org

bullseye (amd64+all)

- essential (amd64+all)
  - reproducible (24)
  - pending (8)

- required (amd64+all)
  - reproducible (31)
    - unreplicable (4)
      - gcc-9.3.6-22
      - apt 2.2.4
      - gcc-10.2.1-6
      - gcc-9.3.0-22
  - retry (1)
  - pending (9)

- build-essential (amd64+all)
  - reproducible (55)
  - unreplicable (28)
in-toto format

```json
{
    "signatures": [
        {
            "keyid": "8deb0bef1d99feb8b9a90fb192ef6d6141641e5c",
            "other_headers": "04000108001d1621048deb0bef1d99feb8b9a90fb192ef6d6141641e5c05026157bb29",
            "signature": "1f0b097baa2b82105ba4054966fa84f35eada6088b218e5856ae8bd7fef04eb37ab2e2c0e8f5d9d7b238704d110"
        }
    ],
    "signed": {
        "name": "rebuild",
        "products": {
            "apt_2.2.4_amd64.deb": {
                "sha256": "75f07c4965ff0813f26623a1164e162538f5e94defba6961347527ed71bc4f3d"
            },
            (...)
        }
    }
}
```

6 https://in-toto.io/
root@debian-11:~# apt-get install tzdata
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be upgraded:
  tzdata
1 upgraded, 0 newly installed, 0 to remove and 842 not upgraded.
Need to get 284 kB of archives.
After this operation, 4,096 B of additional disk space will be used.
Get:1 intoto://deb.debian.org/debian bullseye/main amd64 tzdata all 2021a-1 [284 kB]
(...) 
In-toto verification for '/var/cache/apt/archives/partial/tzdata_2021a-1_all.deb' passed! :) 
Fetched 284 kB in 15s (19.3 kB/s)
Reading changelogs... Done
(...)
Test
(and test again)
tests.reproducible-builds.org

- Continuous tests,
- Archlinux, coreboot, Debian, FreeBSD, NetBSD, OpenWrt, GNU Guix, NixOS, openSUSE, Qubes OS, Yocto Project.
https://tests.reproducible-builds.org

Purpose:

- Build the package,
- Rebuild the package,
- Compare the results,
- This is **not** rebuild.
The second build is different in:

- time,
- file ordering,
- CPU ordering and availability,
- hostname,
- user & group,
- locale,
- kernel,
- etc.
Findings
Identified issues

• Timestamps (recording current time),
• File order,
• (Pseudo-)randomness:
  ○ Temporary file paths,
  ○ UUID,
  ○ Protection against complexity attacks.
Identified issues (cont.)

• CPU and memory related:
  ◦ Code optimizations for current CPU class,
  ◦ Recording of memory addresses,
• Build path,
• Others, e.g. locale settings.
Examples

Timestamps added by build systems
### Timestamps in static libraries

**data.tar**

**./usr/lib/libform_g.a**

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Year</th>
<th>File</th>
<th>Date</th>
<th>Time</th>
<th>Year</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>234 14112</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>f_trace.o</td>
<td>234 14112</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>f_trace.o</td>
</tr>
<tr>
<td>234 14336</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_arg.o</td>
<td>234 14336</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_arg.o</td>
</tr>
<tr>
<td>234 16112</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_attr.o</td>
<td>234 16112</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_attr.o</td>
</tr>
<tr>
<td>234 16368</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_current.o</td>
<td>234 16368</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_current.o</td>
</tr>
<tr>
<td>234 23592</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_def.o</td>
<td>234 23592</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_def.o</td>
</tr>
<tr>
<td>234 14920</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_dup.o</td>
<td>234 14920</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_dup.o</td>
</tr>
<tr>
<td>234 13064</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_ftchoice.o</td>
<td>234 13064</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_ftchoice.o</td>
</tr>
<tr>
<td>234 13992</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_ftlink.o</td>
<td>234 13992</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_ftlink.o</td>
</tr>
<tr>
<td>234 13992</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_info.o</td>
<td>234 13992</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_info.o</td>
</tr>
<tr>
<td>234 13616</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_just.o</td>
<td>234 13616</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_just.o</td>
</tr>
<tr>
<td>234 14688</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_link.o</td>
<td>234 14688</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_link.o</td>
</tr>
<tr>
<td>234 13472</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_max.o</td>
<td>234 13472</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_max.o</td>
</tr>
<tr>
<td>234 13208</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_move.o</td>
<td>234 13208</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_move.o</td>
</tr>
<tr>
<td>234 16296</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_newftyp.o</td>
<td>234 16296</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_newftyp.o</td>
</tr>
<tr>
<td>234 16232</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_opts.o</td>
<td>234 16232</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_opts.o</td>
</tr>
<tr>
<td>234 14312</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_pad.o</td>
<td>234 14312</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_pad.o</td>
</tr>
<tr>
<td>234 13616</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_page.o</td>
<td>234 13616</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_page.o</td>
</tr>
<tr>
<td>234 13504</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_stat.o</td>
<td>234 13504</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_stat.o</td>
</tr>
<tr>
<td>234 14912</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_type.o</td>
<td>234 14912</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_type.o</td>
</tr>
<tr>
<td>234 13488</td>
<td>Jan 2 06:04</td>
<td>2015</td>
<td>fld_user.o</td>
<td>234 13488</td>
<td>Jan 2 06:22</td>
<td>2015</td>
<td>fld_user.o</td>
</tr>
</tbody>
</table>
Timestamps by a template engine

data.tar

```
1  +-- 28 lines: #!/usr/bin/env python...
29 VFSL=valueFromFrameOrSearchList ...
30 VFSL=valueFromSearchList...
31 VFN=valueForName...
32 currentTime=time.time...
33 __CHEETAH_version__ = '2.4.4'
34 __CHEETAH_versionTuple__ = (2, 4, 4, 0)
35 __CHEETAH_genTime__ = 1421397370.77929...
36 __CHEETAH_genTimestamp__ = 'Fri Jan 16 08:45:02 2015'
37 __CHEETAH_src__ = 'dsc.tmpl'
38 __CHEETAH_srcLastModified__ = 'Fri Feb 25 06:40:15 2011'
39 __CHEETAH_docstring__ = 'Autogenerated by Cheetah: The Python...'
40
41 if __CHEETAH_versionTuple__ < RequiredVersionTuple__ < RequiredCheetahVersionTuple:...
42     raise AssertionError('This template was compiled with Cheetah vers...
```

Examples

Archives
Timestamps in gzip headers

```
metadata
1 gzip compressed data, was "br.ispell", last modified: Sat Jan 3 22:47:40 2015
```
## Timestamps in ZIP archives

### data.tar.xz

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date/Time</th>
<th>CRC 32</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>data.tar</td>
<td>1660 bytes</td>
<td>2015 Jan 16 09:55:52</td>
<td>5bb8662c</td>
<td>0775</td>
</tr>
</tbody>
</table>

### ./usr/share/boa-constructor/Docs/boa.apphelp.htb

<table>
<thead>
<tr>
<th>Metadata</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>file last modified on (DOS date/time):</strong></td>
</tr>
<tr>
<td><strong>file last modified on (UT extra field modtime):</strong></td>
</tr>
</tbody>
</table>
## Timestamps in tarballs

### data.tar

### data.tar.xz

### ./usr/share/doc/allegro4-doc/examples/source.tar.gz

### source.tar

<table>
<thead>
<tr>
<th>File</th>
<th>Timestamp</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>MakeLists.txt</td>
<td>2015-01-09 22:32:29</td>
<td>1547</td>
</tr>
<tr>
<td>afinfo.c</td>
<td>2015-01-09 22:32:29</td>
<td>1024</td>
</tr>
<tr>
<td>akaitest.c</td>
<td>2015-01-09 22:32:29</td>
<td>1547</td>
</tr>
<tr>
<td>allegro.pcx</td>
<td>2015-01-09 22:32:29</td>
<td>1547</td>
</tr>
<tr>
<td>digitest.c</td>
<td>2015-01-09 22:32:29</td>
<td>1547</td>
</tr>
<tr>
<td>ex12bit.c</td>
<td>2015-01-09 22:32:29</td>
<td>1547</td>
</tr>
<tr>
<td>ex3buf.c</td>
<td>2015-01-09 22:32:29</td>
<td>1547</td>
</tr>
<tr>
<td>ex3d.c</td>
<td>2015-01-09 22:32:29</td>
<td>1547</td>
</tr>
<tr>
<td>exaccel.c</td>
<td>2015-01-09 22:32:29</td>
<td>1547</td>
</tr>
<tr>
<td>exalpha.c</td>
<td>2015-01-09 22:32:29</td>
<td>1547</td>
</tr>
<tr>
<td>example.dat</td>
<td>2015-01-09 22:32:29</td>
<td>1547</td>
</tr>
<tr>
<td>example.h</td>
<td>2015-01-09 22:32:29</td>
<td>1547</td>
</tr>
<tr>
<td>examples.txt</td>
<td>2015-01-09 22:32:29</td>
<td>1547</td>
</tr>
</tbody>
</table>
Users and groups in tarballs

<table>
<thead>
<tr>
<th>metadata</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 -rw-r--r-- pbuilder1/pbuilder1</td>
</tr>
<tr>
<td>2 -rw-r--r-- pbuilder1/pbuilder1</td>
</tr>
<tr>
<td>3 -rw-r--r-- pbuilder1/pbuilder1</td>
</tr>
<tr>
<td>4 -rw-r--r-- pbuilder1/pbuilder1</td>
</tr>
<tr>
<td>5 -rw-r--r-- pbuilder1/pbuilder1</td>
</tr>
<tr>
<td>6 -rw-r--r-- pbuilder1/pbuilder1</td>
</tr>
<tr>
<td>7 -rw-r--r-- pbuilder1/pbuilder1</td>
</tr>
<tr>
<td>8 -rw-r--r-- pbuilder1/pbuilder1</td>
</tr>
<tr>
<td>9 -rw-r--r-- pbuilder1/pbuilder1</td>
</tr>
<tr>
<td>10 -rw-r--r-- pbuilder1/pbuilder1</td>
</tr>
<tr>
<td>11 -rw-r--r-- pbuilder1/pbuilder1</td>
</tr>
<tr>
<td>12 -rw-r--r-- pbuilder1/pbuilder1</td>
</tr>
</tbody>
</table>
Examples

Timestamps in documentation
Timestamps in TeX output (.dvi)
Examples

“Compiled at/on/by”
Build time via C preprocessor macros

```
80     {"to",1,NULL,'t'},
81     {0,0,0,0}
82     ;
83     
84     #define MSGSIZE 512
85     #define INPUT_TEXT_SIZE_INIT 32
86     
87     char *pgname ="numconv";
88     char compdate[]="Compiled __DATE__ " __TIME__;
89     
90     void ShowNumberSystems(int which) {
```
Build time recorded via Makefile

```bash
149  genversion.$(OBJEXT): $(genversion_SOURCES) genversion.h
150
151  genversion.h: $(top_builddir)/config.status
152      -rm -f @ @.new
153     echo '#define CC "$(CC)"' > @$new
154     echo '#define BUILT_DATE "\"date\"\"' >> @$new
155     echo '#define BUILT_MACH "$(target)"' >> @$new
156     mv @$new @
157
158  version.c: genversion$(EXEEXT)
159      -rm -f version.c
160     ./genversion$(EXEEXT) > version.c
161
162  BUILT_SOURCES += genversion.h version.c
```

package info (click to toggle)

```
amanda 1:3.3.6-4
- links: PTS, VCS
- area: main
- in suites: jessie, sid
- size: 25,248 kB
- ctags: 28,823
- SLOC: ansic: 225,464; perl:
```
Hostname recorded via ./configure

```c
364    VERSION="$VERSION (svn$SVN_REV)"
365    fi
366    fi
367    HOSTNAME=`hostname`
368    DATE=`date +"%d.%m.%Y %H:%M:%S %Z"`
369
370    cat > version.h <<EOF
371    /*
372    * anytun version info
373    *
374    * this file was created automatically
375    * do not edit this file directly
376    * use ./configure instead
377    */
378
379    ifndef ANYTUN_version_h_INCLUDED
380    #define ANYTUN_version_h_INCLUDED
381
382    #define VERSION_STRING_0 " version $VERSION"
383    #define VERSION_STRING_1 "built on $HOSTNAME, $DATE"
```

**Package info (click to toggle)**

*anytun 0.3.5-1*

- links: [PTS](#), [VCS](#)
- area: main
- in suites: jessie, sid
- size: 1,424 kB
- ctags: 1,339
- SLOC: cpp: 9,126; sh: 618; makefile: 367
m4 macros for autoconf (data, build time, username, hostname)

```
AC_FLDIGI_SH_DQ([echo \$ac_configure_args])
ACDEFINE_UNQUOTED([BUILDCONFIGUREARGS], [\$ac_sh_dq], [Configure arguments])
AC_FLDIGI_SH_DQ([date])
ACDEFINE_UNQUOTED([BUILD_DATE], [\$ac_sh_dq], [Build date])
AC_FLDIGI_SH_DQ([whoami])
ACDEFINE_UNQUOTED([BUILD_USER], [\$ac_sh_dq], [Build user])
AC_FLDIGI_SH_DQ([hostname])
ACDEFINE_UNQUOTED([BUILD_HOST], [\$ac_sh_dq], [Build host])
AC_FLDIGI_SH_DQ([\$CXX -v 2>&1 | tail -1])
ACDEFINE_UNQUOTED([BUILD_COMPILER], [\$ac_sh_dq], [Compiler])
```
Examples

File ordering
## File ordering in python-support files

```
./usr/share/python-support/babiloo.private
```

```bash
1 16 lines: /usr/share/babiloo/run.py
17 /usr/share/babiloo/core/modules/utils.py
18 /usr/share/babiloo/core/modules/__init__.py
19 /usr/share/babiloo/core/modules/compression.py
20 /usr/share/babiloo/core/net/xmlrpc.py
21 /usr/share/babiloo/core/net/__init__.py
22 /usr/share/babiloo/core/net/downloader.py
23 /usr/share/babiloo/qt/main_ui.py
24 /usr/share/babiloo/qt/settings.py
25 /usr/share/babiloo/qt/gui_widgets.py
26 /usr/share/babiloo/qt/dictfilemanager.py
27 /usr/share/babiloo/qt/contentSearchModel.py
28 /usr/share/babiloo/qt/historylistmodel.py
29 /usr/share/babiloo/qt/main.py
30 /usr/share/babiloo/qt/abc.py
31 /usr/share/babiloo/qt/onlineDictionary.py
32 /usr/share/babiloo/qt/definitionView.py
33 /usr/share/babiloo/qt/SplashScreen.py
34 /usr/share/babiloo/qt/__init__.py
35 /usr/share/babiloo/qt/Qt2Po.py
36 /usr/share/babiloo/qt/images_rc.py
37 /usr/share/babiloo/qt/about_ui.py
38 /usr/share/babiloo/qt/settings_rc.py
```
Examples

Randomness
Random Python hash order

```python
 OFFSET 201, 10 lines modified
201  self.clearHostsButton.
202  setToolTip(_translate("ChatWidget", "Clear", None))
203  setTitle(_translate("ChatWidget", "Server", None))
204  setText(_translate("ChatWidget", "Port:", None))
205  setToolTip(_translate("ChatWidget", "Enter the server port", None))
206  setToolTip(_translate("ChatWidget", "Shows the status of the server", None))
207  from E5Gui.E5Led import E5Led
208  from E5Gui.E5ComboBox import E5ClearableComboBox
209  from E5Gui.E5LineEdit import E5ClearableLineEdit
201  self.clearHostsButton.
202  setToolTip(_translate("ChatWidget", "Clear", None))
203  self.connectionLed.
204  setTitle(_translate("ChatWidget", "Server", None))
205  setText(_translate("ChatWidget", "Port:", None))
206  setTitle(_translate("ChatWidget", "Shows the connection status", None))
207  self.serverGroup.
```

65 of 73
Examples

Even more timestamps!
# Timestamps in PNG

Even images!

```plaintext
data.tar.xz

data.tar

./usr/share/icons/hicolor/128x128/apps/hedgewars.png

<table>
<thead>
<tr>
<th>sng</th>
<th>sng</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 +--104 lines: #SNG: from stdin-----</td>
<td>1 +--104 lines: #SNG: from stdin-----</td>
</tr>
<tr>
<td>105 ( 64,141,230)</td>
<td>105 ( 64,141,230)</td>
</tr>
<tr>
<td># rgb = (0x40,1</td>
<td># rgb = (0x40,1</td>
</tr>
<tr>
<td>106 }</td>
<td>106 }</td>
</tr>
<tr>
<td>107 bKGD {index: 12}</td>
<td>107 bKGD {index: 12}</td>
</tr>
<tr>
<td>108 tRNS {</td>
<td>108 tRNS {</td>
</tr>
<tr>
<td>109 0}</td>
<td>109 0}</td>
</tr>
<tr>
<td>110 tIME {</td>
<td>110 tIME {</td>
</tr>
<tr>
<td># 31 Dec 2014 05:46:02 GMT</td>
<td># 31 Dec 2014 05:52:27 GMT</td>
</tr>
<tr>
<td>112 year: 2014</td>
<td>112 year: 2014</td>
</tr>
<tr>
<td>113 month: 12</td>
<td>113 month: 12</td>
</tr>
<tr>
<td>114 day: 31</td>
<td>114 day: 31</td>
</tr>
<tr>
<td>115 hour: 5</td>
<td>115 hour: 5</td>
</tr>
<tr>
<td>116 minute: 46</td>
<td>116 minute: 52</td>
</tr>
<tr>
<td>117 second: 2</td>
<td>117 second: 27</td>
</tr>
<tr>
<td>118 }</td>
<td>118 }</td>
</tr>
<tr>
<td>119 tEXT {</td>
<td>119 tEXT {</td>
</tr>
<tr>
<td>120 keyword: &quot;date:create&quot;;</td>
<td>120 keyword: &quot;date:create&quot;;</td>
</tr>
<tr>
<td>121 text: &quot;2014-12-31T05:46:00+00:00&quot;</td>
<td>121 text: &quot;2014-12-31T05:52:25+00:00&quot;</td>
</tr>
<tr>
<td>122 }</td>
<td>122 }</td>
</tr>
</tbody>
</table>
```
Please help!
Please help!

- Do not record time, username, hostname, kernel version...
  - ... or make it optional.
- Sort file paths.
- Sort dictionary keys.
- If you work for a project where we propose patches, please help into merging them!
Help?

- Inventory issues,
- Make packages build reproducibly,
- Fix known common issues:
  - Get reproducible PE binaries,
  - Random filenames with GCC (e.g. annobin),
  - ...
- Debian archive infrastructure
  - Store and distribute *.buildinfo files,
  - ...
- Tools to display local packages reproducibility status (reprotest, diffoscope, etc.).
Summer news

- NSA, CISA, ODNI released Securing the Software Supply Chain: Recommended Practices Guide for Developers\(^7\)

- The document expressly recommends having reproducible builds as part of **advanced** recommended mitigations, along with hermetic builds. Page 31 (page 35 in the PDF) says:

  **Reproducible builds provide additional protection and validation against attempts to compromise build systems.** They ensure the binary products of each build system match: i.e., they are built from the same source, regardless of variable metadata such as the order of input files, timestamps, locales, and paths. (…)

\(^7\)https://media.defense.gov/2022/Sep/01/2003068942/-1/-1/0/ESF_SECURING_THE_SOFTWARE_SUPPLY_CHAIN_DEVELOPERS.PDF
Stay in touch

- Website: https://reproducible-builds.org/,
- Mailing lists: rb-general@lists.reproducible-builds.org,
- Join #reproducible-builds or #debian-reproducible (OFTC).
- https://reproducible-builds.org/events/venice2022/
Thank you for your attention.

Questions? Comments?

https://reproducible-builds.org/