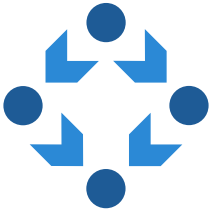
Reproducible Builds

Lunar <lunar@softwareheritage.org>

ESIR3-SI-S9-ASE

2023-11-30



Is compilation a deterministic process?

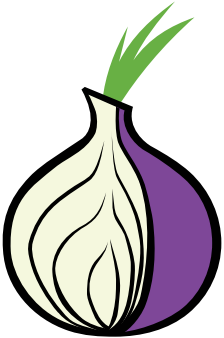
Presentation quiz (1/4)

Presentation quiz (1/4)

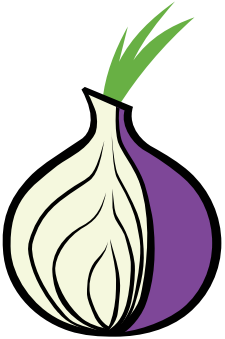
Debian

2007-2016

Presentation quiz (2/4)



Presentation quiz (2/4)

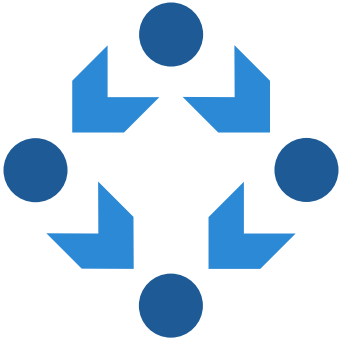


Tor

2009-2014

Presentation quiz (3/4)

Presentation quiz (3/4)



reproducible-builds.org

2013-2016

Presentation quiz (4/4)

Presentation quiz (4/4)

Software Heritage

2022-

Is software development dangerous?

Is software development dangerous?

A tale of three developers…

Alice

Photo by ThisisEngineering RAEng on Unsplash

Photo by the blowup on Unsplash

Bastien

Photo NappyStock

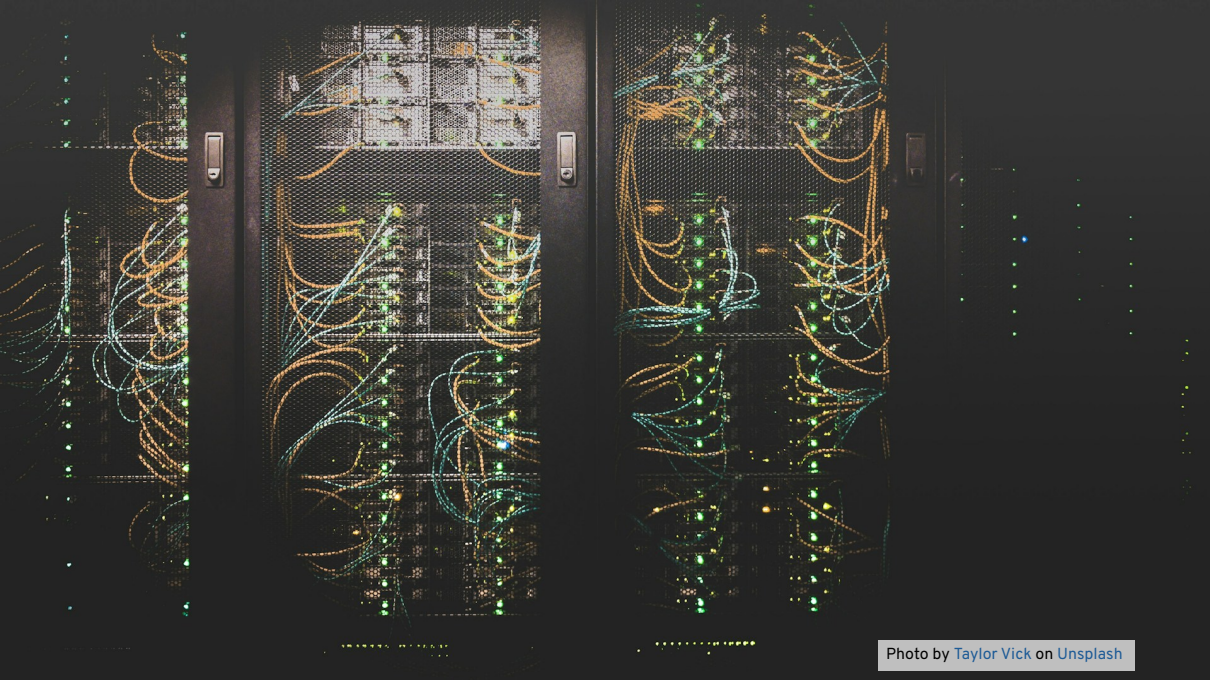


Photo by Taylor Vick on Unsplash

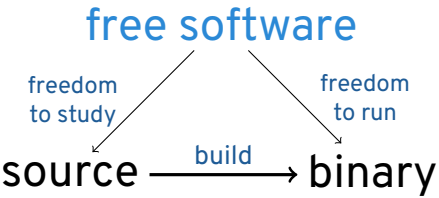
Carole

Photo by Andrew Neel on Unsplash

The problem

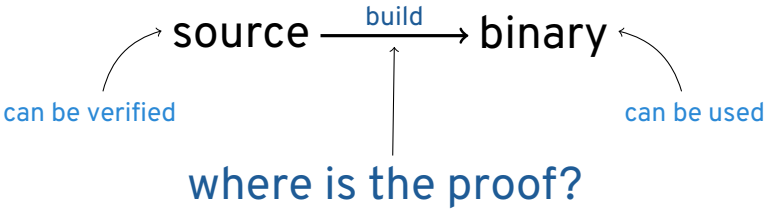
$$\text{source} \xrightarrow{\text{build}} \text{binary}$$

The problem


free software
freedom
to study
freedom
to run
source
build
binary

The problem

The problem


source
build
binary
can be verified
can be used
where is the proof?

The solution

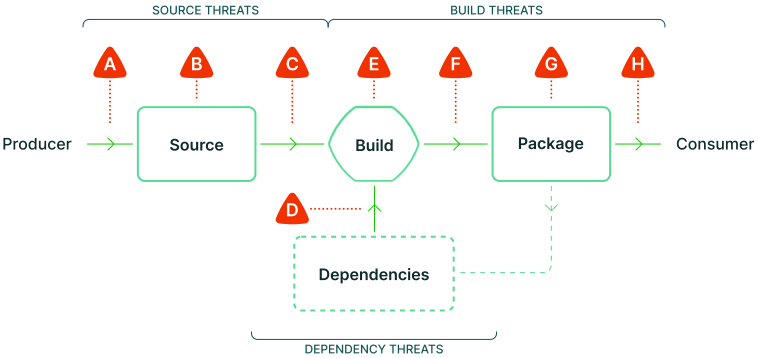
enable anyone to reproduce
identical binary artifacts
from a given source

The solution

We call this:

“reproducible builds”

Supply chain threats overview



SOURCE THREATS

A Submit unauthorized change

B Compromise source repo

C Build from modified source

DEPENDENCY THREATS

D Use compromised dependency

BUILD THREATS

E Compromise build process

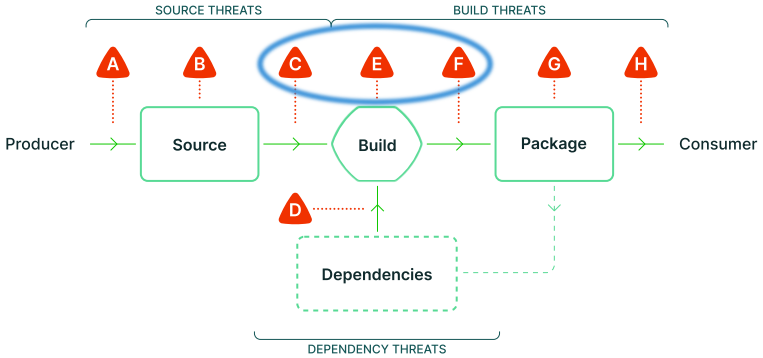
F Upload modified package

G Compromise package registry

H Use compromised package

Source: slsa.dev

Threats we address...



SOURCE THREATS

A Submit unauthorized change

B Compromise source repo

C Build from modified source

DEPENDENCY THREATS

D Use compromised dependency

BUILD THREATS

E Compromise build process

F Upload modified package

G Compromise package registry

H Use compromised package

Source: slsa.dev

These are real threats

At a CIA conference in 2012:

[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.

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

These are real threats

ars TECHNICA

BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE

BIZ & IT —

Apple scrambles after 40 malicious "XcodeGhost" apps haunt App Store

Outbreak may have caused hundreds of millions of people to download malicious apps.

DAN GOODIN - 9/21/2015, 4:40 PM

https://arstechnica.com/information-technology/2015/09/apple-scrambles-after-40-malicious-xcodeghost-apps-haunt-app-store/

These are real threats

webmin

Download Changelog FAQ Documentation Forum

Remote Command Execution [CVE-2019-15231]

- Webmin releases between these versions contain a vulnerability that allows remote command execution!
  Version 1.890 is vulnerable in a default install and should be upgraded immediately - other versions are only vulnerable if changing of expired passwords is enabled, which is not the case by default.
  Either way, upgrading to version 1.930 is strongly recommended. Alternately, if running versions 1.900 to 1.920, edit `/etc/webmin/miniserv.conf`, remove the `passwd_mode=` line, then run `/etc/webmin/restart` command.
  - **More details..**
    Webmin version 1.890 was released with a backdoor that could allow anyone with knowledge of it to execute commands as root. Versions 1.900 to 1.920 also contained a backdoor using similar code, but it was not exploitable in a default Webmin install. Only if the admin had enabled the feature at **Webmin → Webmin Configuration → Authentication** to allow changing of expired passwords could it be used by an attacker.
    Neither of these were accidental bugs - rather, the Webmin source code had been maliciously modified to add a non-obvious vulnerability. It appears that this happened as follows :
    - At some time in April 2018, the Webmin development build server was exploited and a vulnerability added to the `password_change.cgi` script. Because the timestamp on the file was set back, it did not show up in any Git diffs. This was included in the Webmin 1.890 release.

https://www.webmin.com/exploit.html

These are real threats ("SolarWinds hack")

WIKIPEDIA
The Free Encyclopedia

Create account Log in

2020 United States federal government data breach

7 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

In 2020, a major cyberattack suspected to have been committed by a group backed by the Russian government penetrated thousands of organizations globally including multiple parts of the United States federal government, leading to a series of data breaches.[1][28][29] The cyberattack and data breach were reported to be among the worst cyber-espionage incidents ever suffered by the U.S., due to the sensitivity and high profile of the targets and the long duration (eight to nine months) in which the hackers had access.[35] Within days of its discovery, at least 200 organizations around the world had been reported to be affected by the attack, and some of these may also have suffered data breaches.[1][36][37] Affected organizations worldwide included NATO, the U.K. government, the European Parliament, Microsoft and others.[36]

2020 United States federal government data breach

Orion Source Code Replacement

When SUNSPOT finds the Orion solution file path in a running `MsBuild.exe` process, it replaces a source code file in the solution directory, with a malicious variant to inject SUNBURST while Orion is being built. While SUNSPOT supports replacing multiple files, the identified copy only replaces `InventoryManager.cs`.

https://www.crowdstrike.com/blog/sunspot-malware-technical-analysis/

It’s not only about security…

- Minimal diffs on “deliberate” changes
- Cache ratio – save time, money & CO_2
- Find bugs!

Definitions

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 artifacts of a build are the parts of the build results that are the desired primary output.

Relevant attributes of the build environment

Usually include *dependencies and their versions*, *build configuration flags* and *environment variables* as far as they are used by the build system (eg. the locale). *It is preferable to reduce this set of attributes.*

Artifacts

Artifacts would include *executables*, *distribution packages* or *filesystem images*. They would not usually include build logs or similar ancillary outputs.

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

Responsibilities

- Those who write source code:
  Deterministic build system
- Those who provide binaries:
  Reproducible build environment
- Those who distribute binaries:
  Provide a build environment
- Those who verify binaries:
  Perform rebuild and compare results

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

Deterministic build systems

In a nutshell:

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

Common problems for stable inputs

- File order
- Build path
- Parallelism
- Users, groups, `umask`, environment variables, etc.

Common problems for stable outputs

- File order
- Timestamps (recording current time)
- (Pseudo-)randomness:
  - Temporary file paths
  - UUID
  - Protection against complexity attacks (e.g. hashmaps)
- CPU and memory related:
  - Code optimizations for current CPU class
  - Recording of memory addresses
- Locale and timezone settings

Fixing timestamps: `SOURCE_DATE_EPOCH`

- What is it?
  - Environment variable with a reference time
  - Number of seconds since the Epoch (1970-01-01 00:00:00 +0000 UTC)
  - If set, replace “current time of day”
  - Implemented by CMake, gcc, `help2man`, Epydoc, Doxygen, Ghostscript, ocamldoc, sphinx, gettext…
- Set `SOURCE_DATE_EPOCH` in your build system. With Git:

```
SOURCE_DATE_EPOCH=$(git log -1 --pretty=%ct)
```

https://reproducible-builds.org/docs/source-date-epoch/

Volatile inputs can disappear

- Don’t rely on the network
- If you do:
  - Verify content using checksums
  - Have a backup
- The binary distributor should provide a fallback
- For source code, the Software Heritage archive can be used as a fallback
  - See Redoing one paper from ReScience C back on 2020 from the GuixHPC community

Examples

“The unreproducible package” by Bernhard M. Wiedemann:

https://github.com/bmwiedemann/theunreproduciblepackage

Making the build system deterministic

- Perform build *A* in a given environment
- Perform build *B* in an environment as different as possible/desired
- Compare *A* and *B*

Examples variations (from Debian)

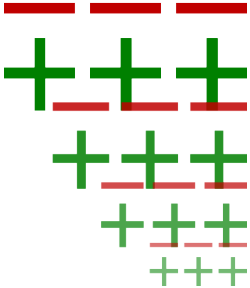
variation	build A	build B
hostname	*hostname*, eg. ionos1-amd64, …	i-capture-the-hostname
domainname	debian.net	i-capture-the-domainname
env TZ	TZ="/usr/share/zoneinfo/Etc/GMT+12"	TZ="/usr/share/zoneinfo/Etc/GMT-14"
env LANG	LANG="C.UTF-8"	LANG="et_EE.UTF-8"
env LANGUAGE	LANGUAGE="en_US:en"	LANGUAGE="et_EE:et"
env LC_ALL	*not set*	LC_ALL="et_EE.UTF-8"
env PATH	PATH="/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:"	PATH="/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/i/capture/the/path"
env BUILDUSERID	BUILDUSERID="1111"	BUILDUSERID="2222"
env BUILDUSERNAME	BUILDUSERNAME="pbuilder1"	BUILDUSERNAME="pbuilder2"
env USER	USER="pbuilder1"	USER="pbuilder2"
env HOME	HOME="/nonexistent/first-build"	HOME="/nonexistent/second-build"
niceness	10	11
uid	uid=1111	uid=2222
gid	gid=1111	gid=2222
/bin/sh	/bin/dash	/bin/bash
build path	/build/a	/build/b
user's login shell	/bin/sh	/bin/bash
user's GECOS	first user, first room, first work-phone, first home-phone, first other	second user, second room, second work-phone, second home-phone, second other
kernel version	6.1.0-13-amd64	6.5.0-0.deb12.1-amd64
umask	0022	0002
CPU type	AMD Opteron 62xx class CPU	Intel(R) Xeon(R) CPU X5550
year, month, date	today or 2025-01-01	398 days difference
filesystem	tmpfs	varied using disorderfs

Debugging problems: diffoscope

- Examines differences **in depth**
- Outputs HTML or plain text showing the differences
- Recursively unpacks archives
- Seeks human readability:
  - uncompresses PDF
  - disassembles binaries
  - unpacks Gettext files
  - *… easy to extend to new file formats*
- Falls back to binary comparison

Available online:

try.diffoscope.org



diffoscope example (HTML output)

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

install.rdf

Offset 5, 15 lines modified		Offset 5, 15 lines modified	
5	<Description about="urn:mozilla:install-manifest">	5	<Description about="urn:mozilla:install-manifest">
6	<em:name>HTTPS-Everywhere</em:name>	6	<em:name>HTTPS-Everywhere</em:name>
7	<em:creator>Mike Perry, Peter Eckersley, & Yan Zhu</em:creator>	7	<em:creator>Mike Perry, Peter Eckersley, & Yan Zhu</em:creator>
8	<em:aboutURL>chrome://https-everywhere/content/about.xul</em:aboutURL>	8	<em:aboutURL>chrome://https-everywhere/content/about.xul</em:aboutURL>
9	<em:id>https-everywhere@eff.org</em:id>	9	<em:id>https-everywhere@eff.org</em:id>
10	<em:type>2</em:type> <!-- type: Extension -->	10	<em:type>2</em:type> <!-- type: Extension -->
11	<em:description>Encrypt the Web! Automatically use HTTPS security on many sites. </em:description>	11	<em:description>Encrypt the Web! Automatically use HTTPS security on many sites. </em:description>
12	<em:version>5.0.6</em:version>	12	<em:version>5.0.7</em:version>
13	<em:multiprocessCompatible>true</em:multiprocessCompatible>	13	<em:multiprocessCompatible>true</em:multiprocessCompatible>

diffoscope example (text output)

```
├── myspell-de-de_20131206-5_all.deb
│   ├── metadata
│   │ @@ -1,3 +1,3 @@
│   │  rw-r--r-- 0/0      4 Jun 11 16:19 2014 debian-binary
│   │ -rw-r--r-- 0/0    775 Jun 11 16:19 2014 control.tar.gz
│   │ +rw-r--r-- 0/0    777 Jun 11 16:19 2014 control.tar.gz
│   │  rw-r--r-- 0/0 325128 Jun 11 16:19 2014 data.tar.xz
│   ├── control.tar.gz
│   │   ├── control.tar
│   │   │   ├── md5sums
│   │   │   │ ⋯ Files in package differs
│   │   │   │
│   │   │
│   ├── data.tar.xz
│   │   ├── data.tar
│   │   │   ├── ./usr/share/hunspell/de_DE.aff
│   │   │   │ @@ -1,11 +1,11 @@
│   │   │   │  # this is the affix file of the de_DE Myspell dictionary
│   │   │   │  # derived from the igerman98 dictionary
│   │   │   │  #
│   │   │   │ -# Version: 20131206 (build 20150801)
│   │   │   │ +# Version: 20131206 (build 20150802)
│   │   │   │  #
│   │   │   │  # Copyright (C) 1998-2011 Bjoern Jacke <bjoern@j3e.de>
│   │   │   │  #
│   │   │   │  # License: GPLv2, GPLv3 or OASIS distribution license agreement
│   │   │   │  # There should be a copy of all of this licenses included
│   │   │   │  # with every distribution of this dictionary. Modified
│   │   │   │  # versions using the GPL may only include the GPL
```

Reprotest

```
# Build a make-based program
$ reprotest "make clean && make" mybinary
```

https://salsa.debian.org/reproducible-builds/reprotest

Recording the environment

For example, Debian `.buildinfo` files:

- Tie in the same file:
  - Sources
  - Generated binaries
  - Packages used to build (with specific version)
- Can be later processed to reinstall environment

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

Example .buildinfo

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

SBOM

SBOM

A software bill of materials (SBOM) declares the inventory of components used to build a software artifact such as a software application. It is analogous to a list of ingredients on food packaging: where you might consult a label to avoid foods that may cause allergies, SBOMs can help organizations or persons avoid consumption of software that could harm them. (Wikipedia)

SBOM

> *A software bill of materials (SBOM) declares the inventory of components used to build a software artifact such as a software application. It is analogous to a list of ingredients on food packaging: where you might consult a label to avoid foods that may cause allergies, SBOMs can help organizations or persons avoid consumption of software that could harm them. (Wikipedia)*

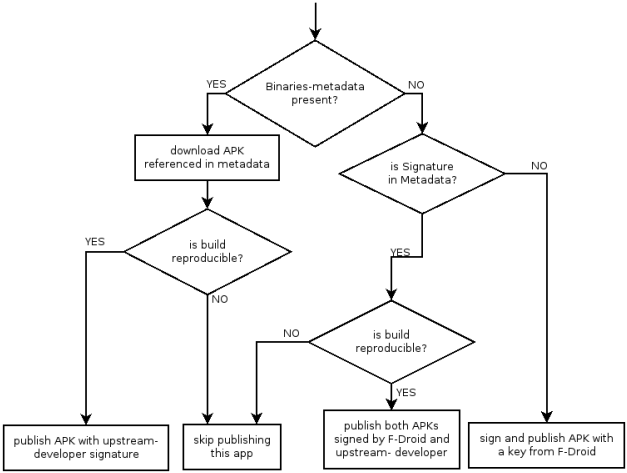
But without reproducible builds, there is no way to verify a SBOM.

How about users?

F-Droid

How about users?

F-Droid



How about users?

```
$ guix challenge \
  --substitute-urls="https://ci.guix.gnu.org https://guix.example.org" \
  openssl git pius coreutils grep
updating substitutes from 'https://ci.guix.gnu.org'... 100.0%
updating substitutes from 'https://guix.example.org'... 100.0%
/gnu/store/…-openssl-1.0.2d contents differ:
  local hash: 0725122r5jnzazaacncwsvp9kgf42266ayyp814v7djxs7nk963q
  https://ci.guix.gnu.org/nar/…-openssl-1.0.2d: 0725122r5jnzazaacncwsvp9kgf42266ayyp814v7djxs7nk963q
  https://guix.example.org/nar/…-openssl-1.0.2d: 1zy4fmaaqcnjrzzajkdn3f5gmjk754b43qkq47llbyak9z0qjyim
  differing files:
    /lib/libcrypto.so.1.1
    /lib/libssl.so.1.1

…

5 store items were analyzed:
  - 2 (40.0%) were identical
  - 3 (60.0%) differed
  - 0 (0.0%) were inconclusive
```

https://guix.gnu.org/en/manual/en/html_node/Invoking-guix-challenge.html

Not a new idea...

```
From: John Gilmore <gnu at cygnus.com>
To: david d `zoo' zuhn <zoo at cygnus.com>
Cc: ian at cygnus.com (Ian Lance Taylor), p3
Subject: Re: comparison results on p3 testing (GNBN)
Date: Tue, 13 Oct 92 11:56:10 -0700

> I think the intention in our tools is to not have the time stamp differ.
> I'm not certain of this though.... anyone else?

I strongly agree that our object files should not contain timestamps.
If you compile the same sources with the same compiler, you should get
the same result -- down to the bit.

    John
```

https://lists.reproducible-builds.org/pipermail/rb-general/2017-January/000309.html

Becoming the norm again

NIST Special Publication 800-218

Secure Software Development Framework (SSDF) Version 1.1:

Recommendations for Mitigating the Risk of Software Vulnerabilities

Murugiah Souppaya
Computer Security Division
Information Technology Laboratory

Karen Scarfone
Scarfone Cybersecurity
Clifton, VA

Donna Dodson*
** Former NIST employee; all work for this publication was done while at NIST.*

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-218

February 2022

https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-218.pdf

Becoming the norm again

Alpine Linux

Apache Maven

Arch Linux

Baserock

Bitcoin Core

BitShares

Buildroot

coreboot

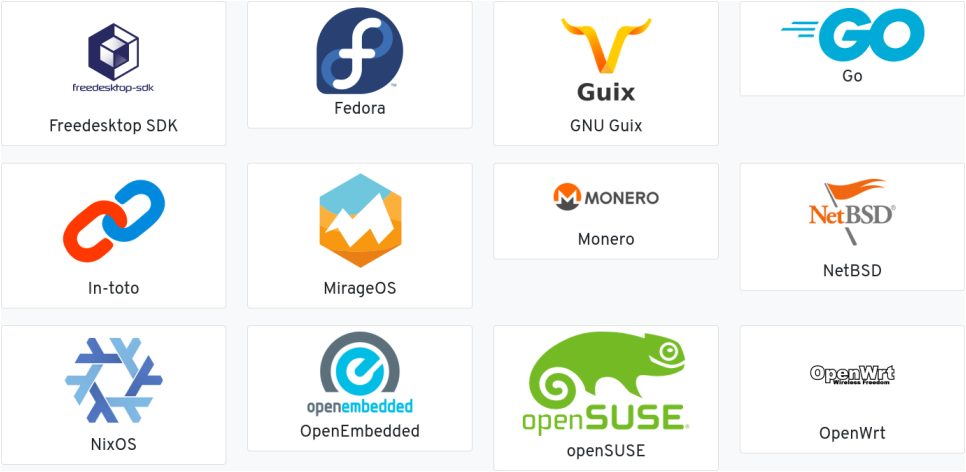
Debian

ElectroBSD

F-Droid

FreeBSD

Becoming the norm again



Becoming the norm again

openEuler

Qubes OS

SecureDrop

Symfony

Tails

Talos Linux

TREZOR

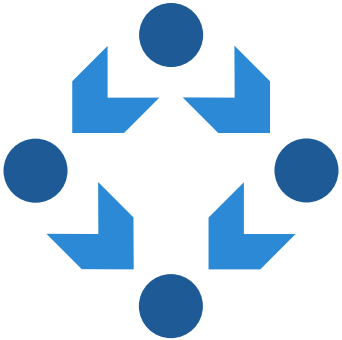
Tor Browser

Webconverger

Yocto Project

Trisquel GNU/Linux

Thanks



reproducible-builds.org