Breaking the Chains of Trusting Trust

Vagrant Cascadian <vagrant@reproducible-builds.org>

FOSSY 2023-07-16

Who am I

	Vagrant
debian user	2001
debian developer	2010
reproducible builds	2015

Trust

Different levels of trust:

- curl `http://example.net/hackme` | sudo sh

Trust

Different levels of trust:

- curl `http://example.net/hackme` | sudo sh
- curl –proto '=https' –tlsv1.2 -sSf `https://sh.rustup.rs` | sh

Trust

Different levels of trust:

- curl `http://example.net/hackme` | sudo sh
- curl –proto '=https' –tlsv1.2 -sSf `https://sh.rustup.rs` | sh
- download file, verify signature ... run code

Trust

Different levels of trust:

- curl `http://example.net/hackme` | sudo sh
- curl –proto '=https' –tlsv1.2 -sSf `https://sh.rustup.rs` | sh
- download file, verify signature ... run code
- download source, verify signature, compile from source

Trust

Different levels of trust:

- curl `http://example.net/hackme` | sudo sh
- curl –proto '=https' –tlsv1.2 -sSf `https://sh.rustup.rs` | sh
- download file, verify signature ... run code
- download source, verify signature, compile from source
- emerge –emptytree @world

Trust

Different levels of trust:

- curl `http://example.net/hackme` | sudo sh
- curl –proto '=https' –tlsv1.2 -sSf `https://sh.rustup.rs` | sh
- download file, verify signature ... run code
- download source, verify signature, compile from source
- emerge –emptytree @world
- rewrite everything in assembly

Trust

Different levels of trust:

- curl `http://example.net/hackme` | sudo sh
- curl –proto '=https' –tlsv1.2 -sSf `https://sh.rustup.rs` | sh
- download file, verify signature ... run code
- download source, verify signature, compile from source
- emerge –emptytree @world
- rewrite everything in assembly
- build it up from transitors

Trust

Different levels of trust:

- curl `http://example.net/hackme` | sudo sh
- curl –proto '=https' –tlsv1.2 -sSf `https://sh.rustup.rs` | sh
- download file, verify signature ... run code
- download source, verify signature, compile from source
- emerge –emptytree @world
- rewrite everything in assembly
- build it up from transitors
- I have a beach, some wood, abundant sunshine, and a lot of time

Trusting Trust

Ken Thompson
Reflections on trusting trust, 1984
`https://archive.org/details/reflections-on-trusting-trust`

The Moral of Trusting Trust

"You can't trust code that you did not totally create yourself.
(Especially code from companies that employ people like me.)
No amount of source-level verification or scrutiny will protect you
from using untrusted code." - Ken Thompson

Did I say 1984, I meant 1974

Karger, 1974
"... insert a trap door into the... compiler...
the trap door can maintain itself,
even when the compiler is recompiled"

Decades of Trust

Since 1974

- 1984: Reflections on trusting trust

Decades of Trust

Since 1974

- 1984: Reflections on trusting trust
- 1980s: some papers about compiling multiple times

Decades of Trust

Since 1974

- 1984: Reflections on trusting trust
- 1980s: some papers about compiling multiple times
- 1990s ... usenet post mumbling about multiple compilers

Decades of Trust

Since 1974

- 1984: Reflections on trusting trust
- 1980s: some papers about compiling multiple times
- 1990s ... usenet post mumbling about multiple compilers
- 2000s: some more papers about compiling multiple times

Decades of Trust

Since 1974

- 1984: Reflections on trusting trust
- 1980s: some papers about compiling multiple times
- 1990s ... usenet post mumbling about multiple compilers
- 2000s: some more papers about compiling multiple times
- 2005: Countering Trusting Trust through Diverse Double-Compiling

Decades of Trust

Since 1974

- 1984: Reflections on trusting trust
- 1980s: some papers about compiling multiple times
- 1990s ... usenet post mumbling about multiple compilers
- 2000s: some more papers about compiling multiple times
- 2005: Countering Trusting Trust through Diverse Double-Compiling
- 2009: Fully Countering Trusting Trust through Diverse Double-Compiling

Decades of Trust

Since 1974

- 1984: Reflections on trusting trust
- 1980s: some papers about compiling multiple times
- 1990s ... usenet post mumbling about multiple compilers
- 2000s: some more papers about compiling multiple times
- 2005: Countering Trusting Trust through Diverse Double-Compiling
- 2009: Fully Countering Trusting Trust through Diverse Double-Compiling
- ... and some high profile compromises!

XcodeGhost or should we say Strawhorse?

XcodeGhost, 2015

- Modified version of Apple's Xcode

XcodeGhost or should we say Strawhorse?

XcodeGhost, 2015

- Modified version of Apple's Xcode
- Over 4000 compromised apps

SolarWhat?

SolarWinds, 2020

- Compromised build server...

SolarWhat?

SolarWinds, 2020

- Compromised build server...
- ...via weak and/or leaked passphrases

SolarWhat?

SolarWinds, 2020

- Compromised build server...
- ...via weak and/or leaked passphrases
- signing certificates compromised

SolarWhat?

SolarWinds, 2020

- Compromised build server...
- ...via weak and/or leaked passphrases
- signing certificates compromised
- possibly 18000 affected installations

The price of Trust

What is the Price...
of Trusting Trust?

Free and Open Source Software

- Use

Free and Open Source Software

Free and Open Source Software

- Use
- Study

Free and Open Source Software

Free and Open Source Software

- Use
- Study
- Change

Free and Open Source Software

Free and Open Source Software

- Use
- Study
- Change
- Share

Free and Open Source Software

Free and Open Source Software

- Use
- Study
- Change
- Share
- Community

Share what exactly

Sharing FOSS...

- source

Share what exactly

Sharing FOSS...

- source
- binaries

Share what exactly

Sharing FOSS...

- source
- binaries
- files packaged for distribution

Where do binaries come from

- Source code...

Where do binaries come from

- Source code...
- Transformed by a toolchain...

Where do binaries come from

- Source code...
- Transformed by a toolchain...
- Into machine code

A taste of source

from bash 5.0 assoc.c:

```
assoc_insert (hash, key, value)
     HASH_TABLE *hash;
     char *key;
     char *value;
{
  BUCKET_CONTENTS *b;

  b = hash_search (key, hash, HASH_CREATE);
  if (b == 0)
    return -1;
  /* If we are overwriting an existing element's value, we're not going to
     use the key.  Nothing in the array assignment code path frees the key
     string, so we can free it here to avoid a memory leak. */
  if (b->key != key)
```

Building the software

```
./configure
make
make install
```

A resulting binary might look like

```
$ head /bin/bash
ELF&@@8 @@88TTTDDPtdDDQtdRtd0<0</lib/ld-linux-aarch64.so.1GNUy;OgUQGNU 04
  #!JzdAPDDB D  @AJ!Ih@i"r
NL@@@AB
0Iq(h   @(
  H &RD!D
    $DP‘
 @A4@ABf L0 dPCDDBE % 32BX@TD$
 @A%

!0‘0@@bBh
 HBH
Xq@ Y       ‘1B
BdH(0"BB1@
```

Reproducible Builds

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

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

Spelling it out

Reproducible Builds provides...

- strong confidence...

Spelling it out

Reproducible Builds provides...

- strong confidence...
- that a binary was produced from a given source...

Spelling it out

Reproducible Builds provides...

- strong confidence...
- that a binary was produced from a given source...
- ...probably!

Once upon a time

- Historically software was reproducible! Every bit counted.

Once upon a time

- Historically software was reproducible! Every bit counted.
- Things eventually got more complicated...

Once upon a time

- Historically software was reproducible! Every bit counted.
- Things eventually got more complicated...
- Bit for bit reproducible GNU toolchain in the early 90s on 10(?) architectures.

Once upon a time

- Historically software was reproducible! Every bit counted.
- Things eventually got more complicated...
- Bit for bit reproducible GNU toolchain in the early 90s on 10(?) architectures.
- And we all forgot.

Once upon a time

- Historically software was reproducible! Every bit counted.
- Things eventually got more complicated...
- Bit for bit reproducible GNU toolchain in the early 90s on 10(?) architectures.
- And we all forgot.
- In 2011 and 2012, Bitcoin and Torbrowser were made reproducible...

Enter Debian

In 2013 folks explore reproducibility for all of Debian

Status in Debian

Debian

- ~34000 source packages

Status in Debian

Debian

- ~34000 source packages
- ~95% reproducible

Status in Debian

Debian

- ~34000 source packages
- ~95% reproducible
- in theory...

Status in Debian

Debian

- ~34000 source packages
- ~95% reproducible
- in theory...
- many submitted patches

Debian: gcc and binutils

gcc and binutils

- test suite logs

Debian: gcc and binutils

gcc and binutils

- test suite logs
- PGO (Profile Guided Optimiziation)

Debian: gcc and binutils

gcc and binutils

- test suite logs
- PGO (Profile Guided Optimiziation)
- LTO (Link Time Optimization)

Debian: linux

linux

- documentation randomness

Debian: linux

linux

- documentation randomness
- other unidentified issues

Debian: linux

linux

- documentation randomness
- other unidentified issues
- fixes available `https://bugs.debian.org/1033663`
  `https://salsa.debian.org/kernel-team/linux/-/merge_requests/741`

Debian: linux

linux

- documentation randomness
- other unidentified issues
- fixes available `https://bugs.debian.org/1033663`
  `https://salsa.debian.org/kernel-team/linux/-/merge_requests/741`
- well, partial fixes, anyways...

Debian: libzstd

libzstd

- recent regression

Status in Guix

GNU Guix

- ~87% reproducible

Status in Guix

GNU Guix

- ~87% reproducible
- in practice!

Status in Guix

GNU Guix

- ~87% reproducible
- in practice!
- 21594 Reproducible!

Status in Guix

GNU Guix

- ~87% reproducible
- in practice!
- 21594 Reproducible!
- 1559 Unreproducible...

Status in Guix

GNU Guix

- ~87% reproducible
- in practice!
- 21594 Reproducible!
- 1559 Unreproducible...
- 1692 Unknown...

Guix made for success

GNU Guix

- reproducible by design

Guix made for success

GNU Guix

- reproducible by design
- normalized build environment

Guix made for success

GNU Guix

- reproducible by design
- normalized build environment
- guix challenge

Guix made for success

GNU Guix

- reproducible by design
- normalized build environment
- guix challenge
- two build farms to compare against

Arch Linux

Arch Linux

`https://reproducible.archlinux.org/`

- ~14000 packages

Arch Linux

Arch Linux
`https://reproducible.archlinux.org/`

- ~14000 packages
- ~86% reproducible

Arch Linux

Arch Linux
`https://reproducible.archlinux.org/`

- ~14000 packages
- ~86% reproducible
- in practice!

But wait, there is more!

- NetBSD 84%

But wait, there is more!

- NetBSD 84%
- OpenWRT 96%-100%

But wait, there is more!

- NetBSD 84%
- OpenWRT 96%-100%
- Coreboot 100%

But wait, there is more!

- NetBSD 84%
- OpenWRT 96%-100%
- Coreboot 100%
- NixOS 95%-99.7%

But wait, there is more!

- NetBSD 84%
- OpenWRT 96%-100%
- Coreboot 100%
- NixOS 95%-99.7%
- Yocto 99.98%

But wait, there is more!

- NetBSD 84%
- OpenWRT 96%-100%
- Coreboot 100%
- NixOS 95%-99.7%
- Yocto 99.98%
- openEuler 96%

But wait, there is more!

- NetBSD 84%
- OpenWRT 96%-100%
- Coreboot 100%
- NixOS 95%-99.7%
- Yocto 99.98%
- openEuler 96%
- openSUSE mostly reproducible (caveats apply)

We miss you!

We once had testing for...

- Alpine
- Fedora

I am not picky about the color of your hat

Wishlist based on current events...

- AlmaLinux
- Rocky Linux

Proof Pudding

Reproducible Builds is totally possible...
...But it only provides one strong link in a chain

Building on a solid foundation of turtles

`https://bootstrappable.org`
Compiling your C compiler with a C compiler
And a C compiler to compile the other C compiler
...Ad infinitum

Java bootstrap

Java bootstrapping

- openjdk17 needs...

Java bootstrap

Java bootstrapping

- openjdk17 needs...
- openjdk16 which needs...

Java bootstrap

Java bootstrapping

- openjdk17 needs...
- openjdk16 which needs...
- ...

Java bootstrap

Java bootstrapping

- openjdk17 needs...
- openjdk16 which needs...
- ...
- openjdk9 ... etc.

Rust bootstrap

Rust bootstrapping

- rust 1.64 needs...

Rust bootstrap

Rust bootstrapping

- rust 1.64 needs...
- rust 1.63 which needs...

Rust bootstrap

Rust bootstrapping

- rust 1.64 needs...
- rust 1.63 which needs...
- ...

Rust bootstrap

Rust bootstrapping

- rust 1.64 needs...
- rust 1.63 which needs...
- ...
- rust 1.54 can be built with mrustc

Rust bootstrap

Rust bootstrapping

- rust 1.64 needs...
- rust 1.63 which needs...
- ...
- rust 1.54 can be built with mrustc
- mrustc is written in C++

Diverse Double Compiling

David A. Wheeler
Fully Countering Trusting Trust through Diverse Double-Compiling, 2009
`https://dwheeler.com/trusting-trust/dissertation/html/wheeler-trusting-trust-ddc.html`

A beautiful Mes

GNU Mes is a Scheme interpreter and C compiler for bootstrapping the GNU System.
`https://www.gnu.org/software/mes/`

We made the same Mes

Bit-for-bit identical Mes built on three different distributions
https://reproducible-builds.org/news/2019/12/21/reproducible-bootstrap-of-mes-c-compiler/

Beginning with a Mes

GNU Guix: The Reduced Binary Seed Bootstrap
`https://guix.gnu.org/en/manual/devel/en/guix.html#Reduced-Binary-Seed-Bootstrap`

- ...

Beginning with a Mes

GNU Guix: The Reduced Binary Seed Bootstrap
`https://guix.gnu.org/en/manual/devel/en/guix.html#Reduced-Binary-Seed-Bootstrap`

- ...
- Reduced to 145MB of bootstrap binaries (from 250MB)

Beginning with a Mes

GNU Guix: The Reduced Binary Seed Bootstrap
`https://guix.gnu.org/en/manual/devel/en/guix.html#Reduced-Binary-Seed-Bootstrap`

- ...
- Reduced to 145MB of bootstrap binaries (from 250MB)
- Using Mes and guile...

Beginning with a Mes

GNU Guix: The Reduced Binary Seed Bootstrap
`https://guix.gnu.org/en/manual/devel/en/guix.html#Reduced-Binary-Seed-Bootstrap`

- ...
- Reduced to 145MB of bootstrap binaries (from 250MB)
- Using Mes and guile...
- Builds from source GCC, binutils, glibc, etc.

Beginning with a Mes

GNU Guix: The Reduced Binary Seed Bootstrap
`https://guix.gnu.org/en/manual/devel/en/guix.html#Reduced-Binary-Seed-Bootstrap`

- ...
- Reduced to 145MB of bootstrap binaries (from 250MB)
- Using Mes and guile...
- Builds from source GCC, binutils, glibc, etc.
- 145MB of binaries is still not really auditable...

Before The Mes and Beyond

GNU Guix: The Full-Source Bootstrap
`https://guix.gnu.org/en/blog/2023/the-full-source-bootstrap-building-from-source-all-the-way-down/`
Now available via guix pull!

- hex0 (357-byte binary)

Before The Mes and Beyond

GNU Guix: The Full-Source Bootstrap
`https://guix.gnu.org/en/blog/2023/the-full-source-bootstrap-building-from-source-all-the-way-down/`
Now available via guix pull!

- hex0 (357-byte binary)
- hex1

Before The Mes and Beyond

GNU Guix: The Full-Source Bootstrap
`https://guix.gnu.org/en/blog/2023/the-full-source-bootstrap-building-from-source-all-the-way-down/`
Now available via guix pull!

- hex0 (357-byte binary)
- hex1
- M0

Before The Mes and Beyond

GNU Guix: The Full-Source Bootstrap
`https://guix.gnu.org/en/blog/2023/the-full-source-bootstrap-building-from-source-all-the-way-down/`
Now available via guix pull!

- hex0 (357-byte binary)
- hex1
- M0
- hex2

Before The Mes and Beyond

GNU Guix: The Full-Source Bootstrap
`https://guix.gnu.org/en/blog/2023/the-full-source-bootstrap-building-from-source-all-the-way-down/`
Now available via guix pull!

- hex0 (357-byte binary)
- hex1
- M0
- hex2
- M1

Before The Mes and Beyond

GNU Guix: The Full-Source Bootstrap
`https://guix.gnu.org/en/blog/2023/the-full-source-bootstrap-building-from-source-all-the-way-down/`
Now available via guix pull!

- hex0 (357-byte binary)
- hex1
- M0
- hex2
- M1
- mescc-tools

Before The Mes and Beyond

GNU Guix: The Full-Source Bootstrap
`https://guix.gnu.org/en/blog/2023/the-full-source-bootstrap-building-from-source-all-the-way-down/`
Now available via guix pull!

- hex0 (357-byte binary)
- hex1
- M0
- hex2
- M1
- mescc-tools
- M2-Planet

Before The Mes and Beyond

GNU Guix: The Full-Source Bootstrap
`https://guix.gnu.org/en/blog/2023/the-full-source-bootstrap-building-from-source-all-the-way-down/`
Now available via guix pull!

- hex0 (357-byte binary)
- hex1
- M0
- hex2
- M1
- mescc-tools
- M2-Planet
- Mes

Before The Mes and Beyond

GNU Guix: The Full-Source Bootstrap
`https://guix.gnu.org/en/blog/2023/the-full-source-bootstrap-building-from-source-all-the-way-down/`
Now available via guix pull!

- hex0 (357-byte binary)
- hex1
- M0
- hex2
- M1
- mescc-tools
- M2-Planet
- Mes
- TinyCC (patched)

Before The Mes and Beyond

GNU Guix: The Full-Source Bootstrap
`https://guix.gnu.org/en/blog/2023/the-full-source-bootstrap-building-from-source-all-the-way-down/`
Now available via guix pull!

- hex0 (357-byte binary)
- hex1
- M0
- hex2
- M1
- mescc-tools
- M2-Planet
- Mes
- TinyCC (patched)
- old versions of GCC, binutils, glibc, gzip, tar ...

Before The Mes and Beyond

GNU Guix: The Full-Source Bootstrap
`https://guix.gnu.org/en/blog/2023/the-full-source-bootstrap-building-from-source-all-the-way-down/`
Now available via guix pull!

- hex0 (357-byte binary)
- hex1
- M0
- hex2
- M1
- mescc-tools
- M2-Planet
- Mes
- TinyCC (patched)
- old versions of GCC, binutils, glibc, gzip, tar ...
- modern GCC and everything

Make it live

`https://github.com/fosslinux/live-bootstrap`

- A live environment

Make it live

`https://github.com/fosslinux/live-bootstrap`

- A live environment
- From kernel and a bit of source code

Make it live

`https://github.com/fosslinux/live-bootstrap`

- A live environment
- From kernel and a bit of source code
- To a reproducibly bootstrapped toolchain

Make it live

`https://github.com/fosslinux/live-bootstrap`

- A live environment
- From kernel and a bit of source code
- To a reproducibly bootstrapped toolchain
- no pregenerated "source" code shortcuts

UEFI based bootstrap

Work-in-progress UEFI bootstrap
`https://git.stikonas.eu/andrius/stage0-uefi`
Only stage0...

Bare Metal Bootstrap

Stage0 on Bare Metal?
`https://git.savannah.nongnu.org/cgit/stage0.git/tree/`

architectures

Full bootstrap only available on x86

. . . x86 toolchain can then cross-compile to x86_64

architectures in progress

- arm
- riscv64
- powerpc64le or powerpc64el

Freedom in your bits and bytes

Free/Libre and Open Source Software
Allows arbitrary third-party verification

Forget Trust, Verify

No need to Trust, All you need is:

- Free/Libre and Open Source Software

Forget Trust, Verify

No need to Trust, All you need is:

- Free/Libre and Open Source Software
- Reproducible Builds

Forget Trust, Verify

No need to Trust, All you need is:

- Free/Libre and Open Source Software
- Reproducible Builds
- Bootstrapping

Forget Trust, Verify

No need to Trust, All you need is:

- Free/Libre and Open Source Software
- Reproducible Builds
- Bootstrapping
- Diverse compilation

Forget Trust, Verify

No need to Trust, All you need is:

- Free/Libre and Open Source Software
- Reproducible Builds
- Bootstrapping
- Diverse compilation
- ... and lots of compile cycles

Copyright and attributions