Where does that code come from?

Git checkout authentication to the rescue of supply chain security

Ludovic Courtès

FOSDEM, 4 February 2023

Inria


Liberating
Dependable
Hackable
10
YEARS
https://guix.gnu.org

```
(define-public hello
  (package
    (name "hello")
    (version "2.12.1")
    (source (origin
              (method url-fetch)
              (uri (string-append "mirror://gnu/hello/hello-"
                                  version ".tar.gz"))
              (sha256 (base32 "0wqd…dz6"))))
    (build-system gnu-build-system)
    (inputs (list gnu-gettext))
    (synopsis "Greetings, FOSDEM!")
    (description "That's what a Guix package looks like.")
    (home-page "https://gnu.org/s/hello")
    (license license:gpl3+)))
```

```
$ guix build hello
```

isolated build: chroot, separate name spaces, etc.

```
$ guix build hello
/gnu/store/ h2g4sf72... -hello-2.12.1
```

hash of **all** the dependencies

```
$ guix build hello
/gnu/store/ h2g4sf72… -hello-2.12.1

$ guix gc --references /gnu/store/…-hello-2.12.1
/gnu/store/…-glibc-2.33
/gnu/store/…-gcc-10.3.0-lib
/gnu/store/…-hello-2.12.1
```

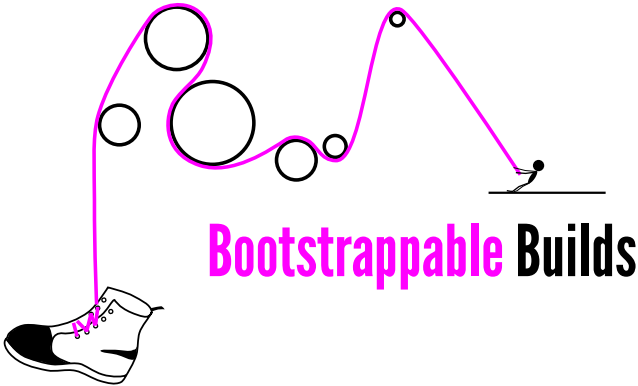
```
$ guix build hello
/gnu/store/ h2g4sf72… -hello-2.12.1

$ guix gc --references /gnu/store/…-hello-2.12.1
/gnu/store/…-glibc-2.33
/gnu/store/…-gcc-10.3.0-lib
/gnu/store/…-hello-2.12.1
```

(nearly) bit-identical for everyone

```
$ guix build hello
/gnu/store/ h2g4sf72… -hello-2.12.1

$ guix gc --references /gnu/store/…-hello-2.12.1
/gnu/store/…-glibc-2.33
/gnu/store/…-gcc-10.3.0-lib
/gnu/store/…-hello-2.12.1
```

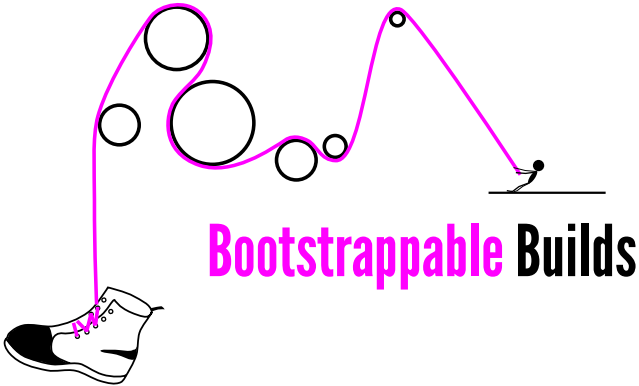
Reproducible Builds

https://reproducible-builds.org



https://bootstrappable.org

→ **“GNU Mes—The Full-Source Bootstrap”**
Jan Nieuwenhuizen, FOSDEM 2021



https://bootstrappable.org

```
$ guix pull
Updating channel ’guix’ from Git repository...
```

The Update Framework

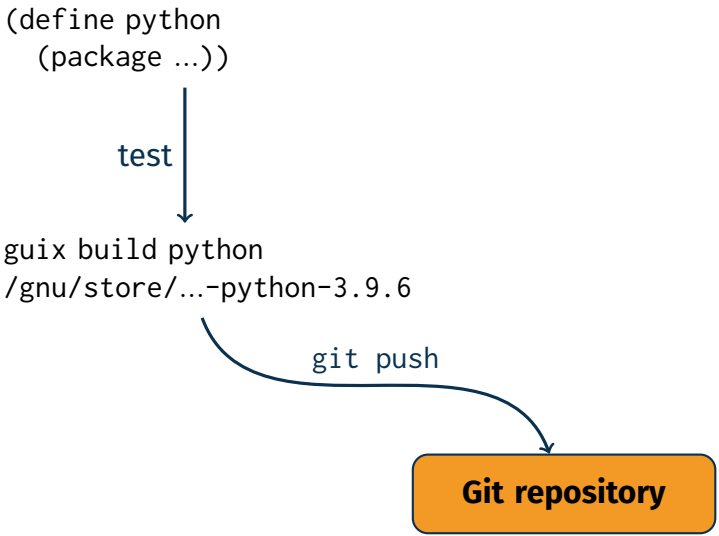
A framework for securing software update systems

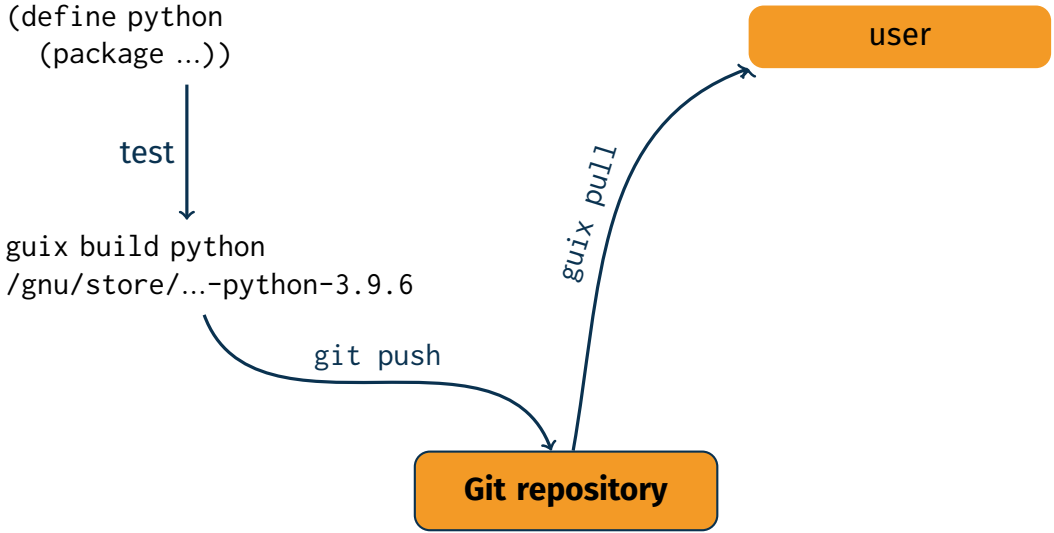
https://theupdateframework.org

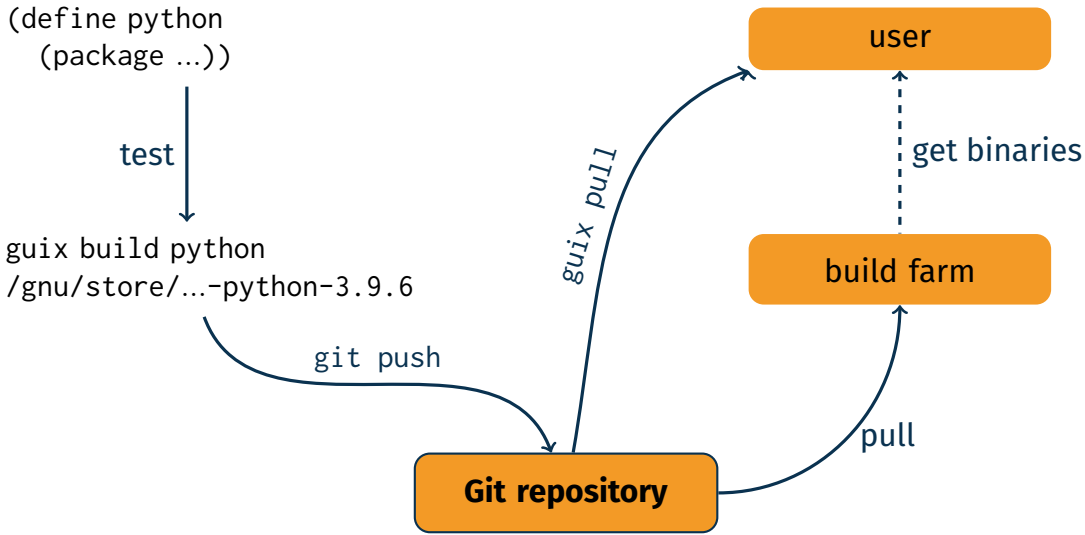
The Update Framework

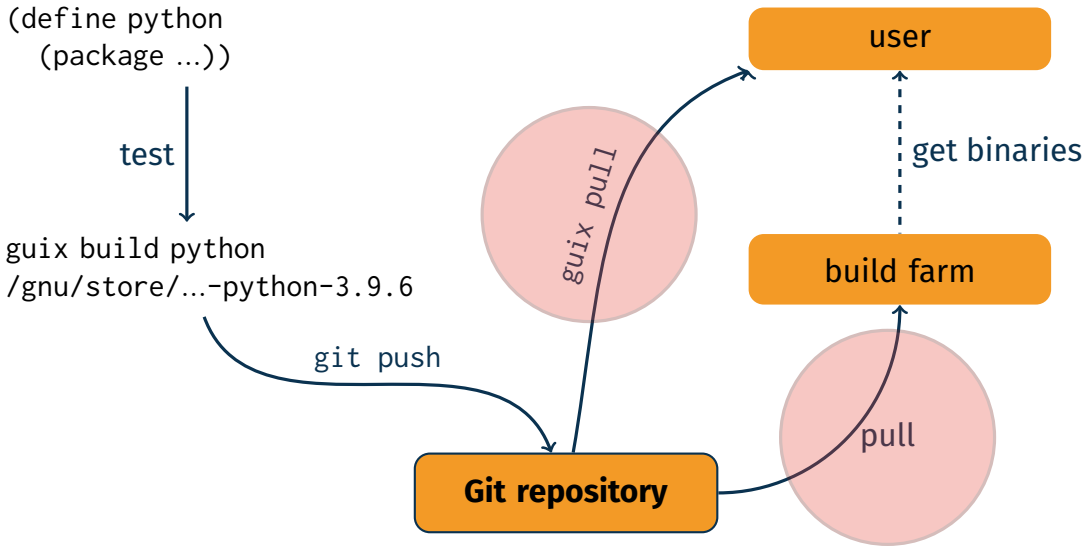
A framework for securing software update systems

https://theupdateframework.org


(define python
(package ...))
test
guix build python
/gnu/store/...-python-3.9.6
git push
Git repository


(define python
(package ...))
test
guix build python
/gnu/store/...-python-3.9.6
git push
Git repository
guix pull
user


(define python
(package ...))
test
guix build python
/gnu/store/...-python-3.9.6
git push
Git repository
guix pull
user
get binaries
build farm
pull


(define python
(package ...))
test
guix build python
/gnu/store/...-python-3.9.6
git push
Git repository
guix pull
user
get binaries
build farm
pull

Commits on Jan 28, 2022

dlib: 19.22 -> 19.23 ···
Ma27 committed 17 hours ago ✓ — Verified — 460c773

live555: add vlc test
jonringer committed 17 hours ago ✓ — 1e8d75b

poke: 1.4 -> 2.0 (#157108) ···
trofi and **SuperSandro2000** committed 18 hours ago ✓ — Partially verified — e87db3c

Merge pull request #156804 from jonringer/python-update-sri-hash
mweinelt committed 18 hours ago ✓ — Verified — 323d853

terraform: fix the plugins wrapper ···
zimbatm authored and **zowoq** committed 18 hours ago ✓ — fe580dc

https://docs.github.com/en/authentication/managing-commit-signature-verification

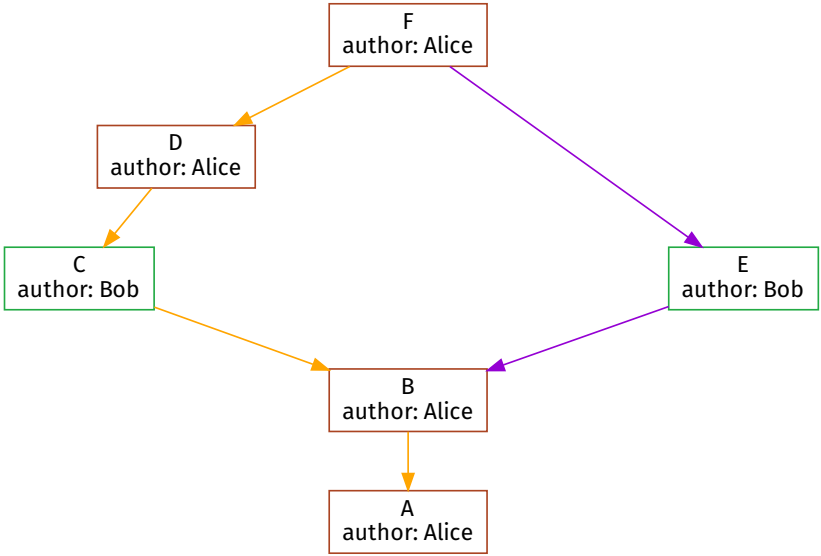
authenticate: *establish the authenticity of something*

authenticity: *undisputed credibility*

— WordNet

- assume **attacker might gain access to the repo**
- protect against **malicious changes**
- … including **downgrade attacks**

- assume **attacker might gain access to the repo**
- protect against **malicious changes**
- … including **downgrade attacks**
- support **off-line authentication**
- support **changing authorizations**

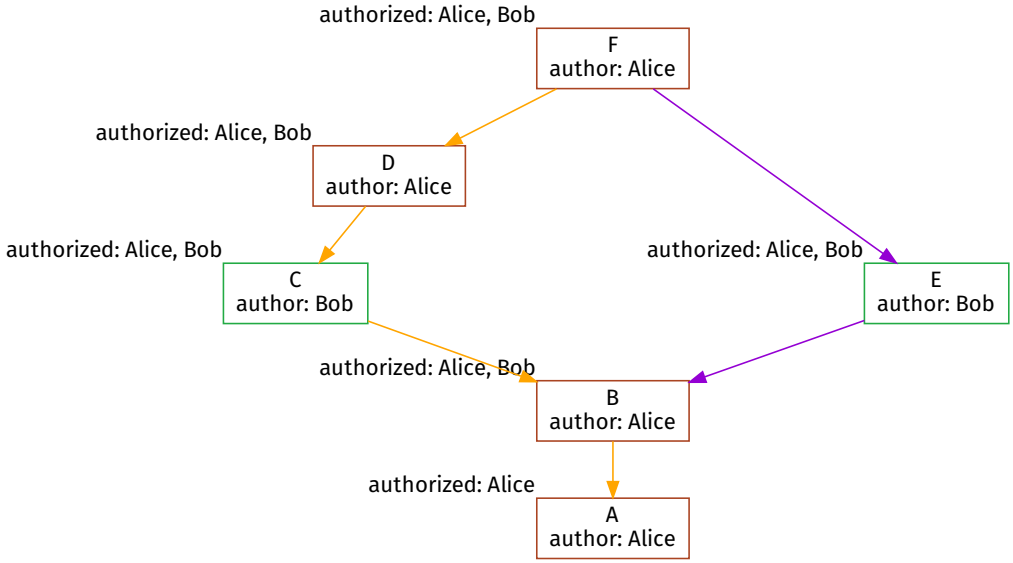

F
author: Alice
D
author: Alice
C
author: Bob
E
author: Bob
B
author: Alice
A
author: Alice

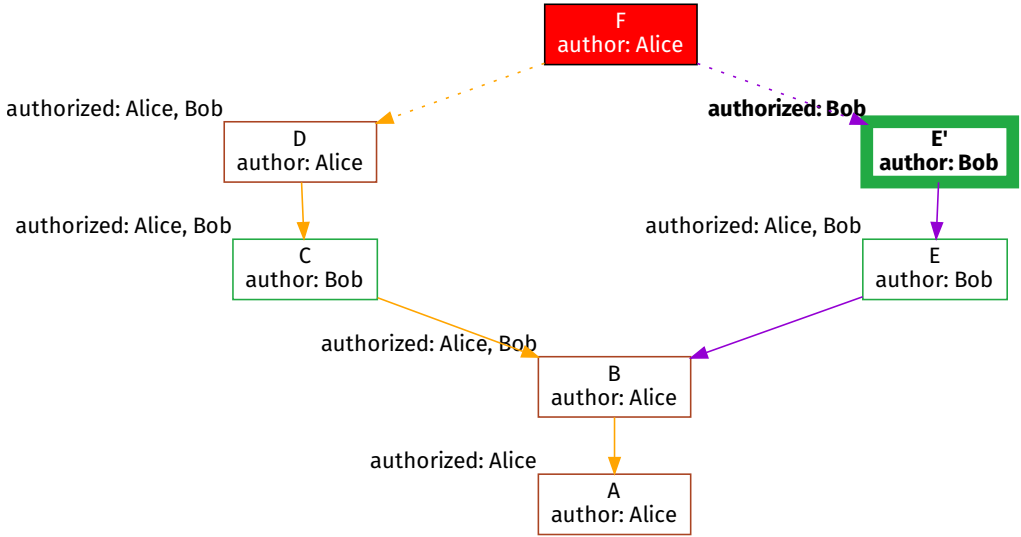
The .guix-authorizations file

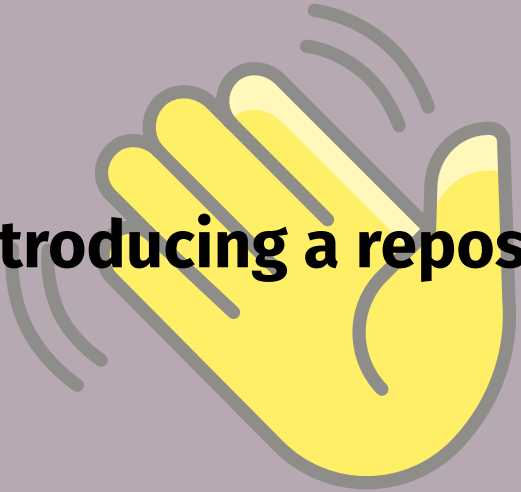
```
( authorizations
  (version 0)

  ;; Authorized committers OpenPGP fingerprints:
  (("AD17 A21E F8AE D8F1 CC02  DBD9 F8AE D8F1 765C 61E3"
    (name "alice"))
  ("2A39 3FFF 68F4 EF7A 3D29  12AF 68F4 EF7A 22FB B2D5"
    (name "bob"))
  ("CABB A931 C0FF EEC6 900D  0CFB 090B 1199 3D9A EBB5"
    (name "charlie"))))
```

Commit is authentic *if and only if* **signed by one of the keys** in the `.guix-authorizations` file of each parent commit.

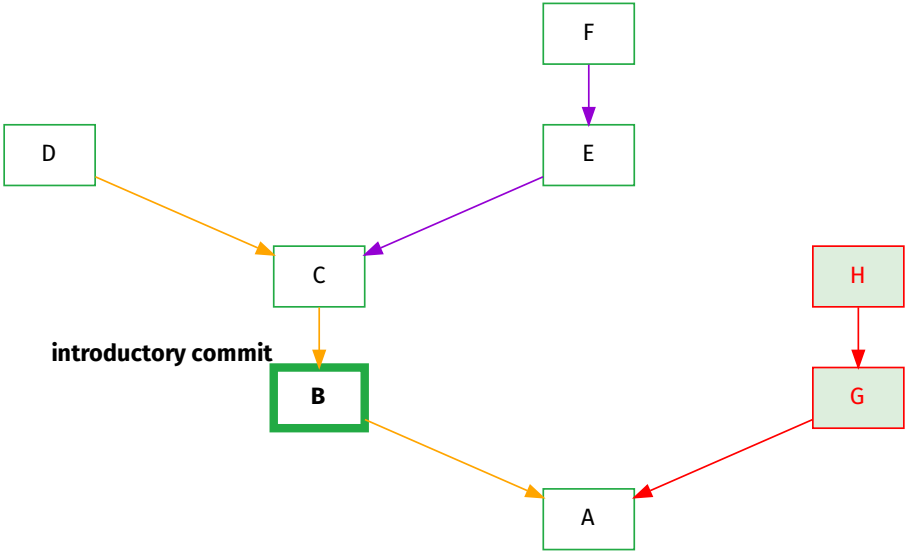
the “authorization invariant”


authorized: Alice, Bob
F
author: Alice
authorized: Alice, Bob
D
author: Alice
authorized: Alice, Bob
C
author: Bob
authorized: Alice, Bob
E
author: Bob
authorized: Alice, Bob
B
author: Alice
authorized: Alice
A
author: Alice


F
author: Alice
authorized: Alice, Bob
D
author: Alice
authorized: Bob
E'
author: Bob
authorized: Alice, Bob
C
author: Bob
authorized: Alice, Bob
E
author: Bob
authorized: Alice, Bob
B
author: Alice
authorized: Alice
A
author: Alice



introducing a repository


F
D
E
C
H
introductory commit
B
G
A

```
(channel
  (name 'my-channel)
  (url "https://example.org/my-channel.git")
  (introduction
   (make-channel-introduction
    "6f0d8cc0d88abb59c324b2990bfee2876016bb86"
    (openpgp-fingerprint
   "CABB A931 C0FF EEC6 900D  0CFB 090B 1199 3D9A EBB5"))))
```

```
$ guix pull
Updating channel 'guix' from Git repository...
Authenticating channel 'guix', 329 new commits...
```

```
$ guix pull --url=https://example.org/mirror.git
Updating channel 'guix' from Git repository...
Authenticating channel 'guix', 329 new commits...
warning: using a mirror, which might be stale
```

```
$ guix pull --url=https://example.org/evil.git
Updating channel 'guix' from Git repository...
Authenticating channel 'guix', 329 new commits...
error: commit c4bba93 not signed by an authorized key
```

```
$ guix git authenticate \
    6f0d8cc0d88abb59c324b2990bfee2876016bb86 \
   "CABB A931 C0FF EEC6 900D  0CFB 090B 1199 3D9A EBB5"
```

```
$ guix git authenticate \
    6f0d8cc0d88abb59c324b2990bfee2876016bb86 \
   "CABB A931 C0FF EEC6 900D  0CFB 090B 1199 3D9A EBB5" \
    --keyring=my-keyring-branch
```

What about downgrade attacks?

```
$ guix describe
  guix cabba9e
    repository URL: https://git.sv.gnu.org/git/guix.git
    commit: cabba9e15900d20927c1f69c6c87d7d2a62040fe
```

```
$ guix describe
  guix cabba9e
    repository URL: https://git.sv.gnu.org/git/guix.git
    commit: cabba9e15900d20927c1f69c6c87d7d2a62040fe

$ guix pull
Updating channel 'guix' from Git repository...
error: commit c0ff33e is not a descendant of cabba9e
```

```
$ guix system describe
  file name: /var/guix/profiles/system-126-link
  label: GNU with Linux-Libre 5.4.15
  bootloader: grub-efi
  channels:
    guix:
      repository URL: https://git.savannah.gnu.org/...
      commit: 93f4511eb0c9b33f5083c2a04f4148e0a494059c
  configuration file: /gnu/store/...-configuration.scm
```

```
$ guix system describe
  file name: /var/guix/profiles/system-126-link
  label: GNU with Linux-Libre 5.4.15
  bootloader: grub-efi
  channels:
    guix:
      repository URL: https://git.savannah.gnu.org/...
      commit: 93f4511eb0c9b33f5083c2a04f4148e0a494059c
  configuration file: /gnu/store/...-configuration.scm

$ guix system reconfigure /etc/config.scm
error: commit c4bba93 is not a descendant of 93f451
```

Wrap-up & outlook.

- **authenticated Git checkouts**
  $\rightarrow$ safe Guix updates!
- **in-band, off-line**: authentication + authorization data is in Git

You can use it on your Git repo!

- **authenticated Git checkouts**
  $\rightarrow$ safe Guix updates!
- **in-band, off-line**: authentication + authorization data is in Git
- protection against **downgrade attacks**
- deployed in Guix **since mid-2020**

https://doi.org/10.22152/programming-journal.org/2023/7/1

Building a Secure Software Supply Chain with GNU Guix

Ludovic Courtès[a]

a Inria, France

Abstract The *software supply chain* is becoming a widespread analogy to designate the series of steps taken to go from source code published by developers to executables running on the users' computers. A security vulnerability in any of these steps puts users at risk, and evidence shows that attacks on the supply chain are becoming more common. The consequences of an attack on the software supply chain can be tragic in a society that relies on many interconnected software systems, and this has led research interest as well as governmental incentives for supply chain security to rise.

GNU Guix is a software deployment tool and software distribution that supports provenance tracking, reproducible builds, and reproducible software environments. Unlike many software distributions, it consists exclusively of source code: it provides a set of package definitions that describe how to build code from source. Together, these properties set it apart from many deployment tools that center on the distribution of binaries.

This paper focuses on one research question: how can Guix and similar systems allow users to securely update their software? Guix source code is distributed using the Git version control system; updating Guix-installed software packages means, first, updating the local copy of the Guix source code. Prior work on secure software updates focuses on systems very different from Guix—systems such as Debian, Fedora, or PyPI where updating consists in fetching metadata about the latest binary artifacts available—and is largely inapplicable in the context of Guix. By contrast, the main threats for Guix are attacks on its *source code repository*, which could lead users to run inauthentic code or to downgrade their system. Deployment tools that more closely resemble Guix, from Nix to Portage, either lack secure update mechanisms or suffer from shortcomings.

Our main contribution is a model and tool to authenticate new Git revisions. We further show how, building on Git semantics, we build protections against downgrade attacks and related threats. We explain implementation choices. This work has been deployed in production two years ago, giving us insight on its actual use at scale every day. The Git checkout authentication at its core is applicable beyond the specific use case of Guix, and we think it could benefit to developer teams that use Git.

As attacks on the software supply chain appear, security research is now looking at every link of the supply

Unified deployment toolbox vs. patchwork

- **end-to-end integration** vs. “artifact flow”
- **verifiability** vs. attestation
- **commit graph** vs. version strings
- …

From source code
to deployed binaries:
provenance tracking
& verifiability are the key.

Guix

https://guix.gnu.org/

ludo@gnu.org | @civodul@toot.aquilenet.fr