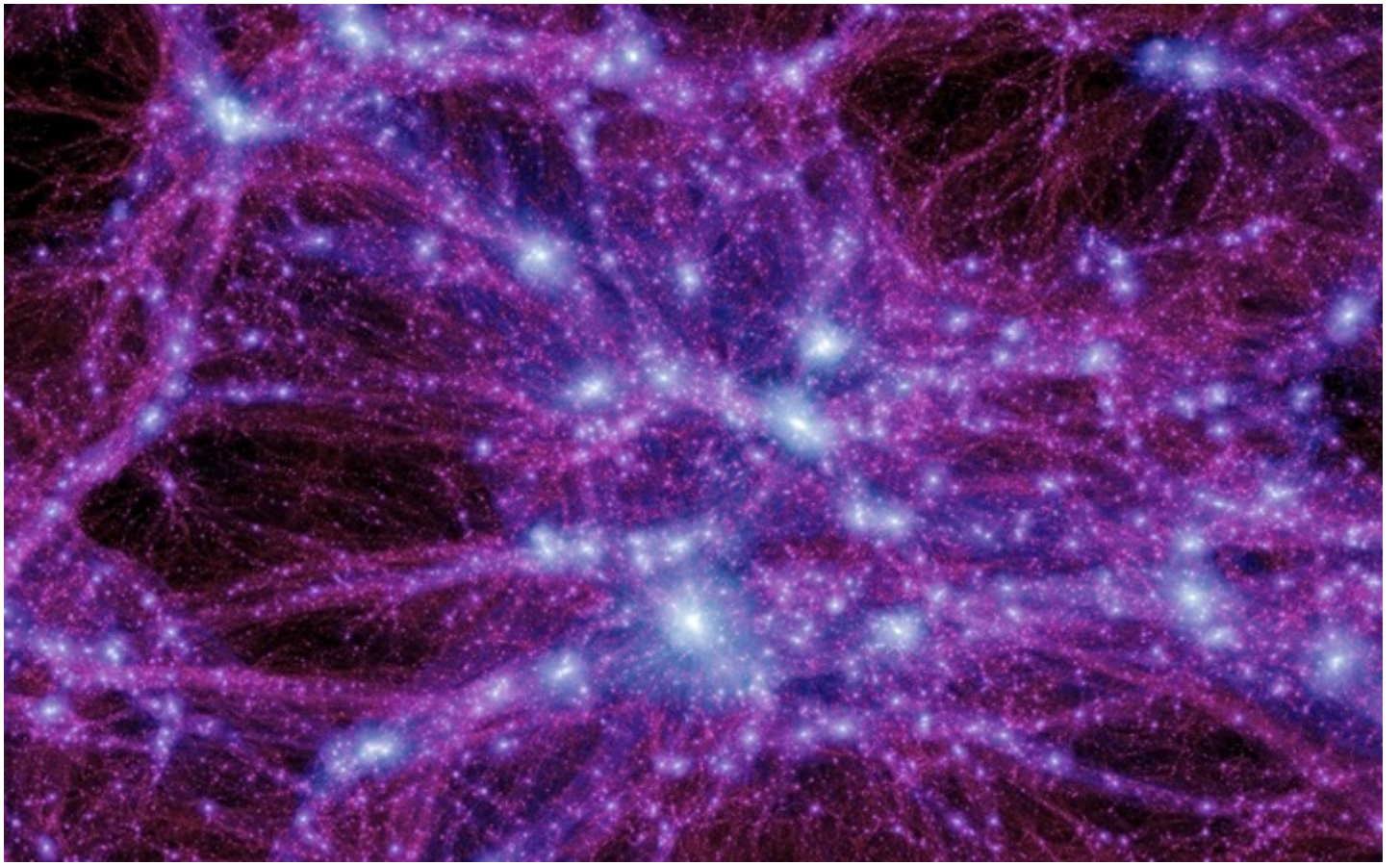
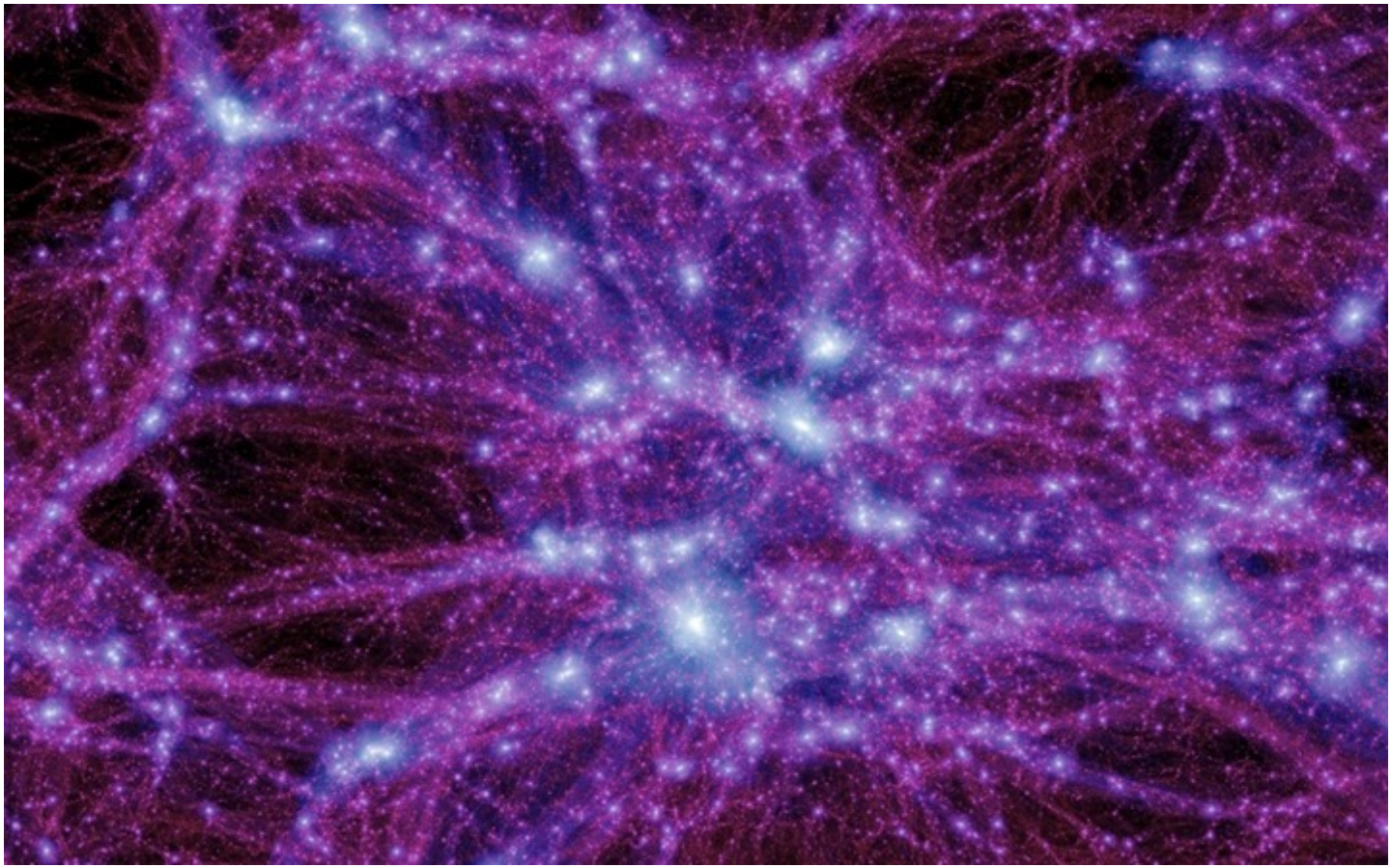
Reproducibly building artifacts that contain embedded signatures

Martin Schwaighofer

Dependency Graph

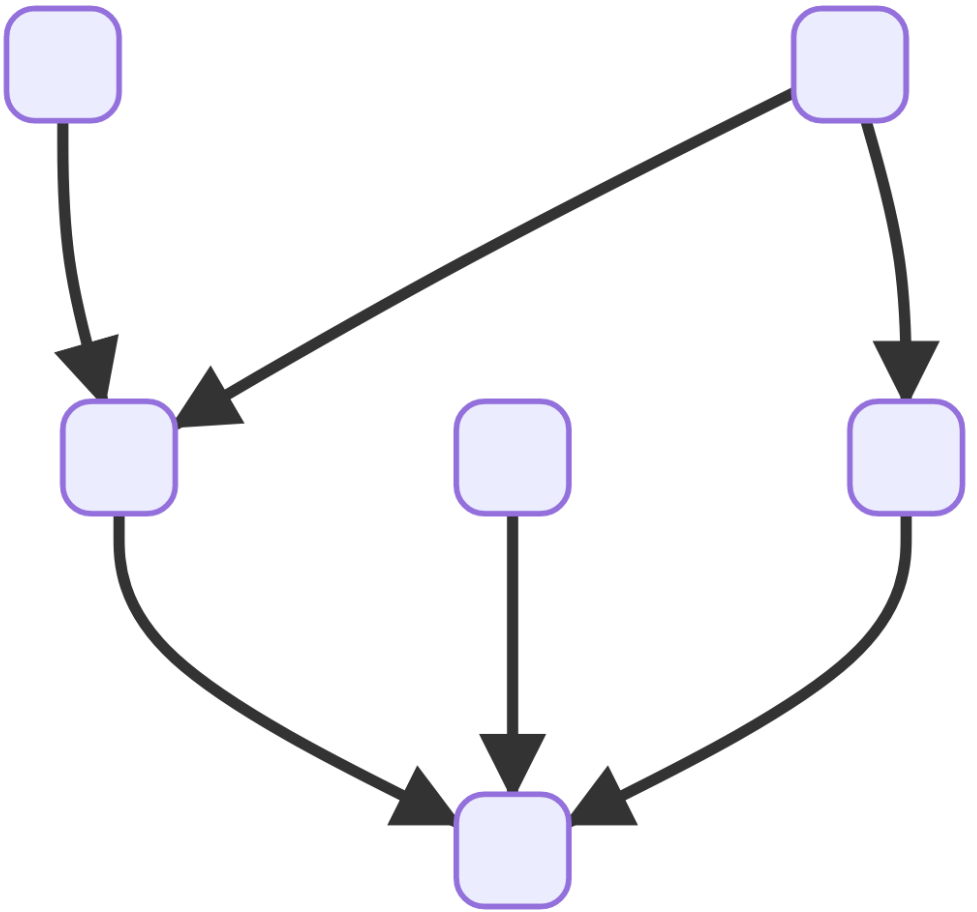


Dependency Graph



- https://commons.wikimedia.org/wiki/File:Dark_matter.jpg

Dependency Graph



Small hint about tooling

- `nix build --rebuild`
  - Re-runs individual build steps

Small hint about tooling

- `nix build --rebuild`
  - Re-runs individual build steps

Issues with reproducing signatures

- they might be anywhere
  - nested deeply in the final artifact

Small hint about tooling

- `nix build --rebuild`
  - Re-runs individual build steps

Issues with reproducing signatures

- they might be anywhere
  - nested deeply in the final artifact
- Signature scheme might not be deterministic

Small hint about tooling

- `nix build --rebuild`
  - Re-runs individual build steps

Issues with reproducing signatures

- they might be anywhere
  - nested deeply in the final artifact
- Signature scheme might not be deterministic
- **We might not have access to the key**

Small hint about tooling

- `nix build --rebuild`
  - Re-runs individual build steps

Issues with reproducing signatures

- they might be anywhere
  - nested deeply in the final artifact
- Signature scheme might not be deterministic
- **We might not have access to the key**

If we can't avoid the signature,

Small hint about tooling

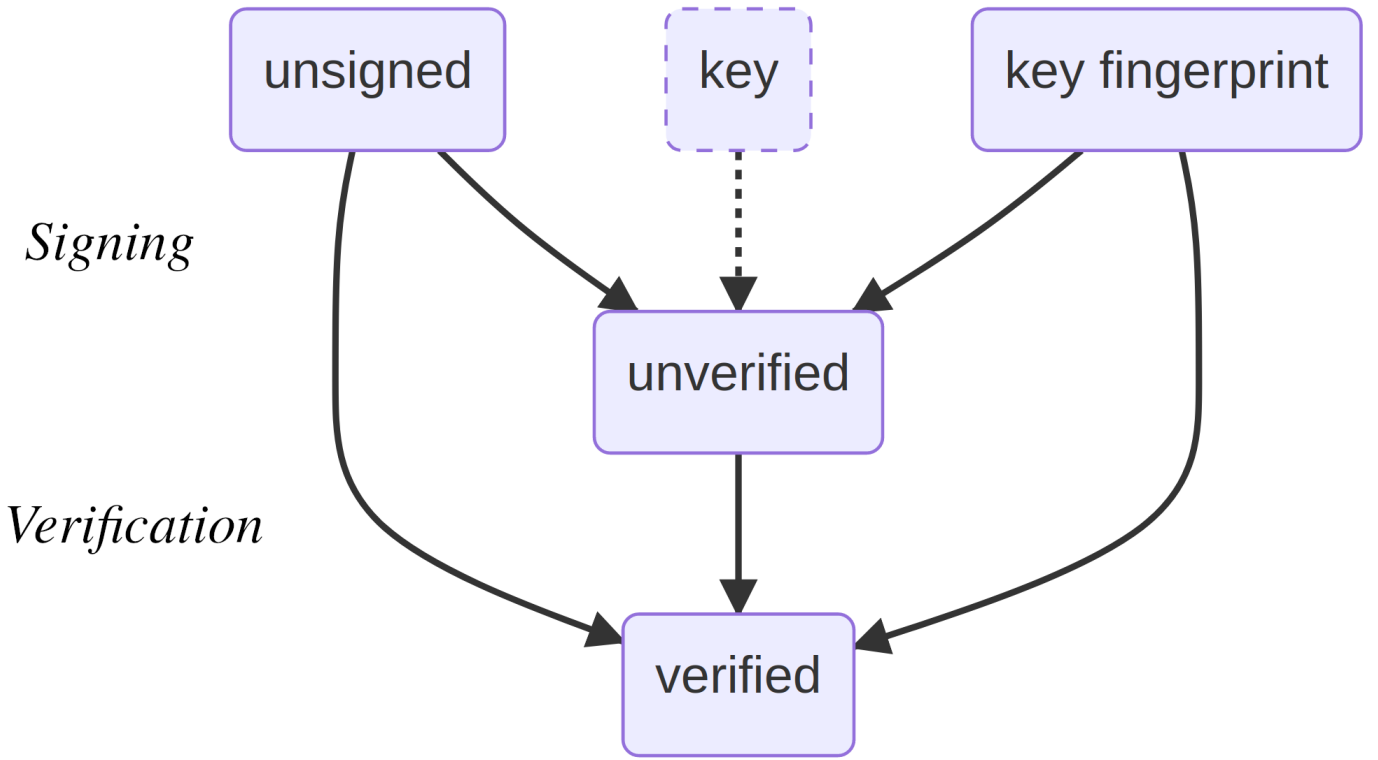
- `nix build --rebuild`
  - Re-runs individual build steps

Issues with reproducing signatures

- they might be anywhere
  - nested deeply in the final artifact
- Signature scheme might not be deterministic
- **We might not have access to the key**

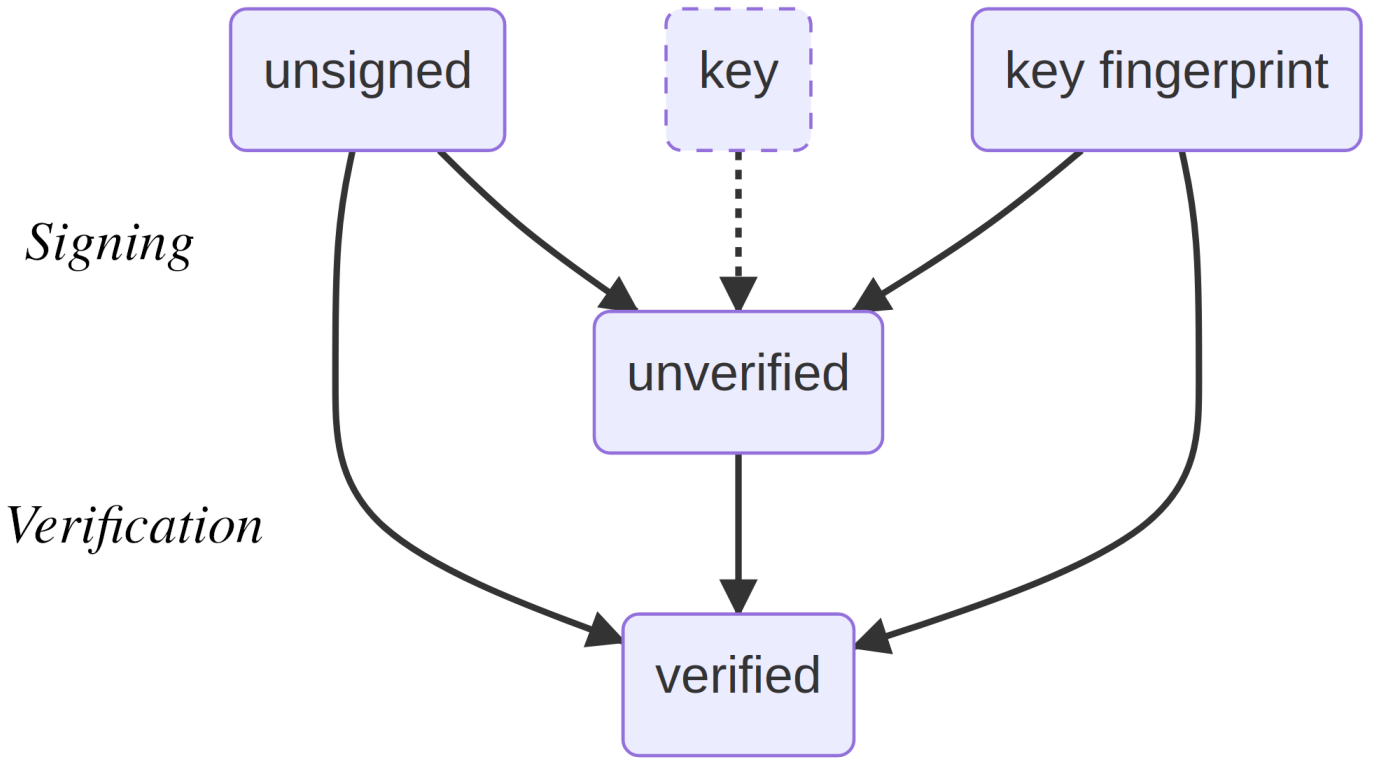
If we can't avoid the signature,
let's manage the problem and verify the signature instead!

Rough Idea



- One derivation for signing, one derivation for verification.

Rough Idea



- One derivation for signing, one derivation for verification.
- Reproducing verification makes it unecessary to trust the signing derivation.

Signing

```
let
  unverified = pkgs.runCommandLocal "sign-apk" {
    buildInputs = [ pkgs.apksigner ];
  } ''
    mkdir -p $out
    cp ${unsigned}/app-unsigned.apk .
    apksigner sign --ks ${keystore-location} [...] \
                                        app-unsigned.apk

    cp app-unsigned.apk $out/app-signed.apk
```

Signing

```
let
  unverified = pkgs.runCommandLocal "sign-apk" {
    buildInputs = [ pkgs.apksigner ];
  } ''
    mkdir -p $out
    cp ${unsigned}/app-unsigned.apk .
    apksigner sign --ks ${keystore-location} [...] \
                                        app-unsigned.apk

    cp app-unsigned.apk $out/app-signed.apk

    # load bearing comment:
    # ${key-fingerprint}
    # makes derivation depend on key
```

Signing

```
let
  unverified = pkgs.runCommandLocal "sign-apk" {
    buildInputs = [ pkgs.apksigner ];
  } ''
    mkdir -p $out
    cp ${unsigned}/app-unsigned.apk .
    apksigner sign --ks ${keystore-location} [...] \
                                        app-unsigned.apk

    cp app-unsigned.apk $out/app-signed.apk

    # load bearing comment:
    # ${key-fingerprint}
    # makes derivation depend on key

    # TODO:
    # * verify key fingerprint matches signature we produced
    #   - so we can't upload the wrong thing to a substituter
    '';
...
```

Verifying

```
let
  verified = pkgs.runCommandLocal "verify-apk" {
    buildInputs = [ pkgs.apksigner ];
    VERIFIES = unverified;
  } ''
    mkdir -p $out
    keyfp=${key-fingerprint}
    apksigner verify --print-certs \
              ${unverified}/app-signed.apk \
                          | tee signatures.log

    cat signatures.log | grep SHA-256 | grep $keyfp
    echo "signed with $keyfp"
    cp ${unverified}/app-signed.apk $out/app-signed.apk
```

Verifying

```
let
  verified = pkgs.runCommandLocal "verify-apk" {
    buildInputs = [ pkgs.apksigner ];
    VERIFIES = unverified;
  } ''
    mkdir -p $out
    keyfp=${key-fingerprint}
    apksigner verify --print-certs \
              ${unverified}/app-signed.apk \
                          | tee signatures.log

    cat signatures.log | grep SHA-256 | grep $keyfp
    echo "signed with $keyfp"
    cp ${unverified}/app-signed.apk $out/app-signed.apk

    # TODO:
    # * verify unsigned artifact matches expectation
    '';
...
```

Adapted r13y.com tool to display results

193 out of 194 (99.48%) paths in tl packages.x86_64-linux.default inst image are reproducible!

1 unchecked

---

unreproduced paths

---

unchecked paths

- `/nix/store/jxqdp2rrzgiqxy4i1dkvvgl2m9ym4fgw-impure.drv` (**verified** by `/nix/store/szl9x1br2lr1gjasviknz8v4m2k74nkn-verify.drv`)

- https://github.com/mschwaig/r13y.com/tree/use-nix-command
- That code doesn't really work outside a demo yet.

Thanks and please say hello 👋

- Martin Schwaighofer
- PhD student with René Mayrhofer at
  - Institute of Networks and Security
  - Johannes Kepler University Linz
- 📩 martin.schwaighofer@ins.jku.at
- 👾 https://github.com/mschwaig
- 🐦 https://twitter.com/mschwaig
- Researching reproducibility and its applications
- Looking for feedback and collaborators

Bonus slides

Simple Opinionated AOSP builds

Simple Opinionated AOSP builds by an external Party (SOAP)

The SOAP project aims to build AOSP in a reproducible manner and identify differences to the reference builds provided by Google. As reference builds we track a selection of the following:

- Factory images for phones by Google
- Generic system images as provided by Android CI

The project enables the broader Android community, as well as any interested third parties, to track differences differences between AOSP and official Google builds. Furthermore, it can act as basis for future changes improving the reproduceability of Android.

Reproducible builds in general

Reproducible builds in general have been widely recognized as an important step for improving trust in executable binaries. More general information on reproducible builds can be found at . More specifically for Android, increased reproduceability bridges the gap between source code provided by the AOSP and the factory images running on millions of Google phones today.

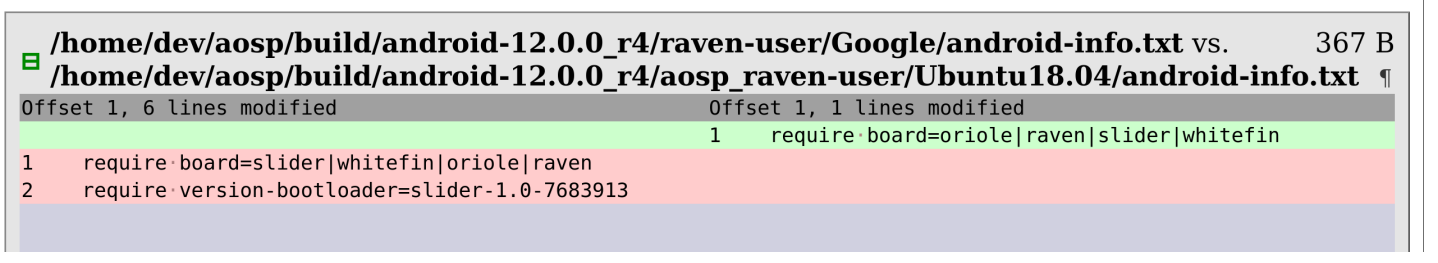
- Web: https://android.ins.jku.at/reproducible-builds/
- Paper: https://dl.acm.org/doi/10.1145/3507657.3528537

What is `diffoscope`

Input:

```
diffoscope --html file1.img file2.img
```

Output:

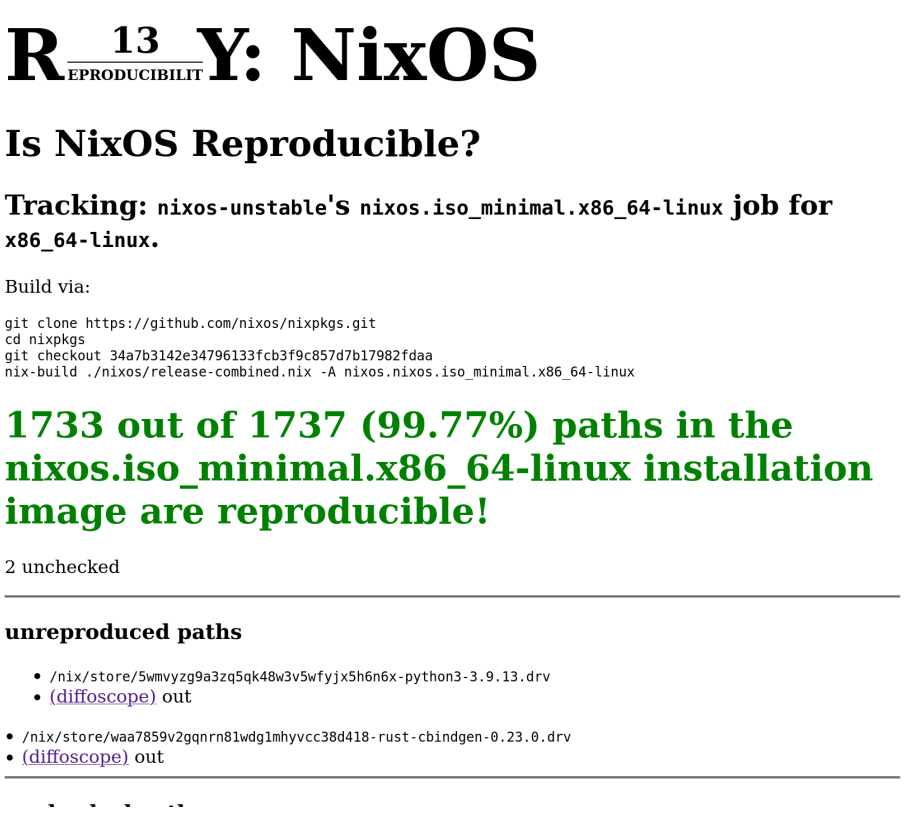


/home/dev/aosp/build/android-12.0.0_r4/raven-user/Google/android-info.txt vs. 367 B
/home/dev/aosp/build/android-12.0.0_r4/aosp_raven-user/Ubuntu18.04/android-info.txt ¶

Offset 1, 6 lines modified	Offset 1, 1 lines modified
	1 require board=oriole|raven|slider|whitefin
1 require board=slider|whitefin|oriole|raven	
2 require version-bootloader=slider-1.0-7683913	

Example Source: https://android.ins.jku.at/soap/android-12.0.0_r4_raven-user_Google__android-12.0.0_r4_aosp_raven-user_docker-Ubuntu18.04/android-info.txt.diffoscope.html-dir/index.html

What is r13y.com



R13Y: NixOS

Is NixOS Reproducible?

Tracking: `nixos-unstable`'s `nixos.iso_minimal.x86_64-linux` job for `x86_64-linux`.

Build via:

```
git clone https://github.com/nixos/nixpkgs.git
cd nixpkgs
git checkout 34a7b3142e34796133fcb3f9c857d7b17982fdaa
nix-build ./nixos/release-combined.nix -A nixos.nixos.iso_minimal.x86_64-linux
```

1733 out of 1737 (99.77%) paths in the nixos.iso_minimal.x86_64-linux installation image are reproducible!

2 unchecked

---

unreproduced paths

- `/nix/store/5wmvyzg9a3zq5qk48w3v5wfyjx5h6n6x-python3-3.9.13.drv`
- (diffoscope) out

- `/nix/store/waa7859v2gqnrn81wdg1mhyvcc38d418-rust-cbindgen-0.23.0.drv`
- (diffoscope) out

- Website: https://r13y.com/
- Tool that generates it: https://github.com/grahamc/r13y.com

Acknowledgements

This work has been carried out within the scope of Digidow, the Christian Doppler Laboratory for Private Digital Authentication in the Physical World. We gratefully acknowledge financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, the Christian Doppler Research Association, 3 Banken IT GmbH, ekey biometric systems GmbH, Kepler Universitätsklinikum GmbH, NXP Semiconductors Austria GmbH & Co KG, and Österreichische Staatsdruckerei GmbH.