Software Transparency:
package security beyond signatures and reproducible builds

DebConf 19 Curitiba, 21 July 2019

Benjamin Hof
hof@cs.tum.edu
Introduction

Design
Introduction

Design
Motivation

Lots of reports on compromised update systems

- so many reports of compromised update systems e.g., Asus 2018
- Chris Lamb: A tale of three developers

Propose add-on security feature on top of Secure APT and reproducible builds

1. architecture overview
2. report progress
Software distribution in APT
Current approach

uploader → archive → CDN

signing key

packages, meta data

client
Software distribution in APT

Security model

uploader → archive → CDN

signing key

packages, meta data

client
Software distribution in APT

Security model

uploader -> archive -> CDN

- signing key
- packages, meta data

client
Introduction

Design
Goals

1. cryptographically ensure everybody sees the same InRelease
   ⇒ detect targeted backdoors!
2. analysis results applicable to all installations
   ⇒ validate reproducible builds for every installation!
3. cryptographic attribution of misbehaviour
Design
Add a log server

1. analyse log operation and
2. log elements (e.g. test reproducible builds)

1. inclusion of InRelease in log
2. consistency: log history was not manipulated

archive
meta data
source code
log
CDN
client auditor
Design
Software Transparency architecture for APT

1. analyse log operation and
2. log elements (e.g. test reproducible builds)

1. inclusion of InRelease in log
2. consistency: log history was not manipulated
Authenticating meta data and source code
Hash tree over a list of items: Merkle tree
Authenticating meta data and source code

Hash tree over a list of items: Merkle tree

Log can efficiently and cryptographically prove:

- inclusion of a given element in the list
- append-only operation of the list

⇒ no need to trust log, can be verified
Progress made

- ansible repository
- new log implementation: https://strans.net.in.tum.de/strans/v0/
  - distribute proofs over mirror network
  - compatible to log “hubs”
  - versioned data structures
- dak patches (wip)
- secured mirror: https://burgundy.net.in.tum.de/debian/

todo: APT, monitor

(previous work: replayed two years of stretch mostly during testing on software prototypes)
Up next

- BoF on Monday, 1430, in Sala de Videoconferencia
- rb-general@lists.reproducible-builds.org
- code now on salsa:
  https://salsa.debian.org/reproducible-builds/transparency
Summary

- detection of targeted backdoors
- forensic auditability
  - inspectable source code for every binary
  - verified mapping between source and binary
  - identify maintainer responsible for distribution
  - proof attributing misbehaviour

Questions?