Transparency for Control Plane Software

Benjamin Hof and Georg Carle

Friday 13th October, 2017

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich
Outline

Motivation

Design

Discussion
Motivation

Goal

Make sure that we install same controller software as everybody else

- analysis carries over
- no targeted backdoors
Motivation
Software distribution process

software update

SDN controller

untrusted

trusted
Motivation

Software distribution process

- Archive
- CDN
- Software update
- SDN controller
Motivation

Software distribution process
Motivation

Software distribution process
Motivation

Securing multi-file software projects [1, 3]

release file:

expires: 2017-10-13 12:00
name: ryu
depends: python-2.7
source hash: 0xa4bc3...
binary hash: 0xb98ff...

⇒ secure release file
Design

Hash tree over a list of items: Merkle tree

- tree root authenticates leaves
Design
Merkle tree-based transparency logs [2]

Log server operates Merkle tree:

- hash tree over list of *release files*

Log can efficiently and cryptographically prove:

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

Verifying log operations

- Archive
- Release file
- CDN
- SDN controller
- Log
- Auditor

Verify log inclusion of release file
Design

Verifying log operations

auditor and monitor verify the correct operation of the log

- inclusion of elements
- append-only
Discussion

Motivation

Design

Discussion
Discussion

Properties

**log**: untrusted, but kept trustworthy through verification by auditor and monitor

**monitor**: inspects elements
Discussion
Extending this idea

When source code is logged, monitors can:

- verify uploader signature on source code
- verify relationship between binary and source code
- detect targeted backdoors

Implementation for APT/Debian

1. prototype
2. deployment can start locally with slightly adjusted architecture
