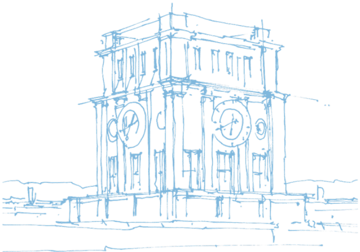
Transparency for Control Plane Software

Benjamin Hof and Georg Carle

Friday 13th October, 2017

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



Outline

Motivation

Motivation

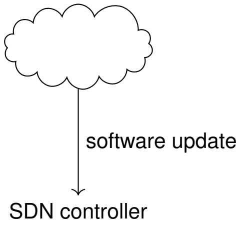
Goal

Make sure that we install same controller software as everybody else

- analysis carries over
- no targeted backdoors

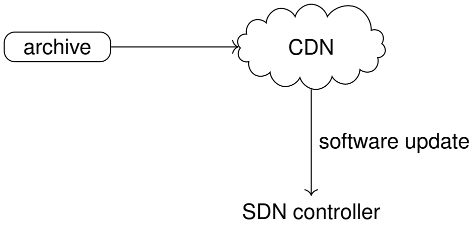
Motivation

Software distribution process



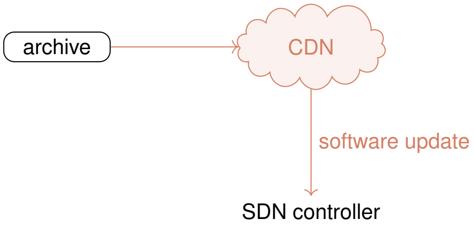
Motivation

Software distribution process



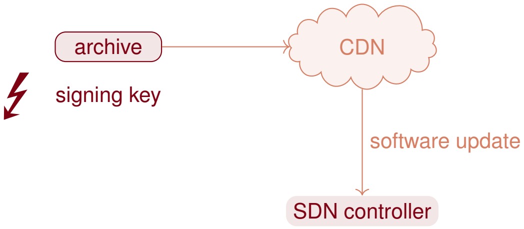
Motivation

Software distribution process



Motivation

Software distribution process



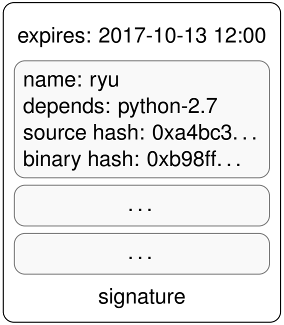
□ untrusted

■ trusted

Motivation

Securing multi-file software projects [1, 3]

release file:

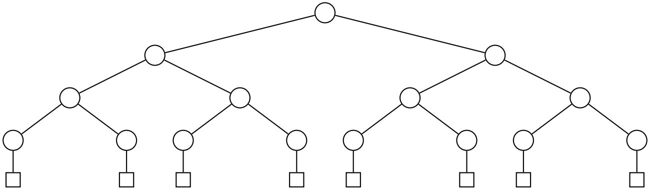


$\Rightarrow$ secure release file

Design

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:

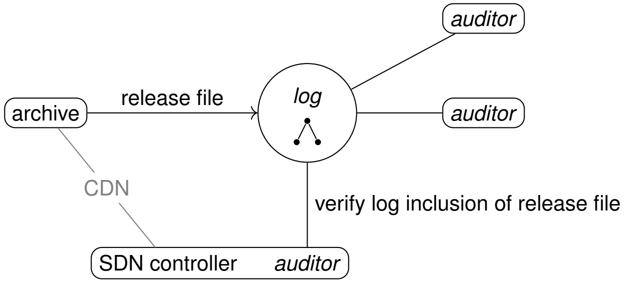
- hash tree over list of *release files*

Log can efficiently and cryptographically prove:

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

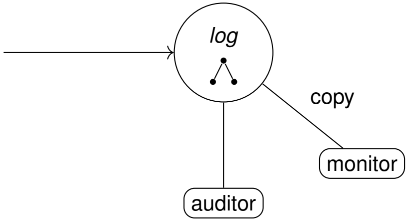
Design

Verifying log operations



Design

Verifying log operations



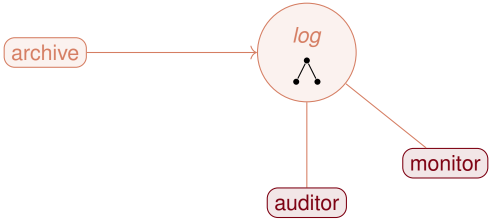
auditor and monitor verify the correct operation of the log

- inclusion of elements
- append-only

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

Bibliography

[1] J. Cappos, J. Samuel, S. Baker, and J. H. Hartman.
A Look in the Mirror: Attacks on Package Managers.
In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 565–574. ACM.

[2] M. Chase and S. Meiklejohn.
Transparency Overlays and Applications.
In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 168–179. ACM.

[3] J. Samuel, N. Mathewson, J. Cappos, and R. Dingledine.
Survivable Key Compromise in Software Update Systems.
In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 61–72. ACM.