Formal Verification of Secure Forwarding Protocols

Tobias Klenze, Christoph Sprenger, David Basin

CSF'21
June 2021

ETH Zürich

Proofs available!

Contact: tobias.klenze@inf.ethz.ch
The Internet lacks network security

Autonomous Systems
(e.g., Internet Service Providers)

To: Bob

Alice

Bob
Path-aware Internet

To: Bob

Alice

Bob

A

D

C

E

F
Path-aware Internet

Balance Control

Alice

$$$

$$$
Path-aware Internet

Two parts: ① Routing (*creating & authorising paths*), …
Path-aware Internet

Two parts: 1. Routing (*creating & authorising paths*), …
Path-aware Internet

Two parts: ① Routing (creating & authorising paths), ② Forwarding (using paths)
Path-aware Internet

Two parts: ① Routing (*creating & authorising paths*), ② Forwarding (*using paths*)

Path Authorization:
Packets traverse the network only along authorized paths.
Challenges for the Verification of Path Authorization

Challenge #1

Arbitrary, **unbounded** set of authorized paths, and unbounded path length.

Challenge #2

**Expressiveness** to formulate path authorization.

Challenge #3

**Large class** of protocols.
Verification of Path Authorization

Our approach: Refinement in Isabelle/HOL.

Arbitrary, unbounded set of authorized paths, unbounded path length.

Abstract
- prove path authorization

Concrete
- prove refinement

No attacker
- Distributed, colluding Dolev-Yao attacker
- Cryptographic authenticators

No authenticators

Environment parameter
Contributions:

- Proving security of a class of forwarding protocols
- Insights into protocol class
- Low-effort proofs: Eight instances, only static reasoning, not about transitions
In \(1\), paths are created: one Hop Field \(HF_i = \langle \delta_i, \sigma_i \rangle\) per node \(i\).

- \(\delta_i\): local forwarding information
- \(\sigma_i\): authenticator (e.g., MAC)

In \(2\), Alice embeds a path.

In \(2\), routers check validity of authenticator.

\[
\begin{align*}
HF_C &= \langle \delta_C, \sigma_C \rangle \\
HF_B &= \langle \delta_B, \sigma_B \rangle \\
HF_A &= \langle \delta_A, \sigma_A \rangle
\end{align*}
\]
How to define the authenticator?

\[ \sigma_i = MAC_{Key(i)} \langle \delta_i \rangle \]

Authenticating local \( \delta \) is not enough!
Authenticators must protect subsequent path

\[ \sigma_i = \text{MAC}_{\text{Key}(i)} \langle \delta_i, \sigma_{i+1} \rangle \]

⊥ for last hop field

\[ \sigma_A = \text{MAC}_{\text{Key}(A)} \langle \delta_A, \sigma_B \rangle \]
\[ \sigma_A = \text{MAC}_{\text{Key}(A)} \langle \delta_A, \text{MAC}_{\text{Key}(B)} \langle \delta_B, \sigma_C \rangle \rangle \]
\[ \sigma_A = \text{MAC}_{\text{Key}(A)} \langle \delta_A, \text{MAC}_{\text{Key}(B)} \langle \delta_B, \text{MAC}_{\text{Key}(C)} \langle \delta_C, \bot \rangle \rangle \rangle \]

\[ \text{extract}(\sigma_A) = [\delta_A, \delta_B, \delta_C] \]
Authenticators must protect subsequent path

\[ \sigma_i = \text{Cryptographic check} \]

\[ \sigma_A = \text{MAC}_{\text{Key}(A)} \langle \delta_A, \text{MAC}_{\text{Key}(B)} \langle \delta_B, \text{MAC}_{\text{Key}(C)} \langle \delta_C, \perp \rangle \rangle \rangle \]

extract(\sigma_A) = [\delta_A, \delta_B, \delta_C]

Parametrized Concrete Model

- Three protocol parameters
- Five static conditions
Conclusion

Three verification challenges:

- Arbitrary, unbounded sets of authorized paths
- Expressiveness for path authorization
- Low effort proofs for new protocol variants

We solved these challenges via refinement and parametrization in Isabelle/HOL.

Future work: Whole Internet architectures to verify!

Contact: tobias.klenze@inf.ethz.ch