Verified Secure Routing

David Basin
ETH Zurich

EPFL, Summer Research Institute
June 2017
Team Members

Verification Team

Information Security
David Basin
Tobias Klenze
Ralf Sasse
Christoph Sprenger
Thilo Weghorn

Programming Methodology
Marco Eilers
Peter Müller

Scion Design & Development Team

Network Security
Samuel Hitz
Adrian Perrig
Motivation and Context

Routing problems with the status quo (inter-AS routing)
Routing between autonomous systems

• Network of networks run by different institutions

• Nodes correspond to **Autonomous Systems (ASes)**
  - Set of *routers* run by common institution (Telcos, ISPs, companies)
  - 50,000+ ASes, e.g., your typical university or large corporation.
Autonomous systems and routers

• Multiple paths between ASes: 2,1,4 and 2,3,4
• Computed in background by Border Gateway Protocol (BGP) and just one will be selected and used to configure routers
Path between two ASes computed using Border Gateway Protocol (BGP)

Path: 8,7,5,4,2,1
Traffic flow: 1,2,4,5,7,8
• ASes exchange reachability information (*paths*)

• Policies programmed by network operators
  - Decisions on what is accepted, rejected, or propagated
  - Any AS can announce any address range it wants

• It is all based on trust! Motivations may vary!
Who controls the Internet?

- Control over paths is completely distributed
  - Border Gateway Protocol (BGP): all nodes flood path announcements

- No inbound traffic control
Who controls Internet paths?

Traceroute Path 4: from Chicago, IL to Tehran, Iran

Source: Renesys Path Measurements
Three concrete examples

Pakistan DoS against Youtube (2 hours, 2008)

Ukraine ISP hijacks UK routes including UK Atomic Weapons

Fribourg’s government address space stolen for 3 days by SPAMers
Scion

Routing as it should be
Scion Project
Secure Future Internet Architecture

- Design & Implementation, 75+ man years
- Design of routing / forwarding protocols, support ecosystem, and numerous extensions
- Clean slate, yet compatible with existing Internet
- Not just a research prototype: Growing deployment on 5 continents, 4 ISDs, 26 ASes
SCION Overview

- Isolation Domains (ISD)
- Control Plane: routing
  - Path exploration
  - Path registration
  - Path resolution
- Data Plane: packet forwarding
SCION Isolation Domain (ISD)

(1) Agreement:
Each region agrees on a common trust root.

(2) Failure Isolation:
No ISD can influence another ISD’s control plane.
SCION Routing (Control Plane)

Routing Phases:
(1) Path Exploration
(2) Path Registration
(3) Path Resolution

- Path Construction Beacons (PCB) are Sequence of signed Hop Fields
- Hop Fields (HF) carry the routing information for one AS
SCION Routing (Control Plane)

Routing Phases:
(1) Path Exploration
(2) Path Registration
(3) Path Resolution

- **Path Construction Beacons (PCB)** are Sequence of signed Hop Fields
- **Hop Fields (HF)** carry the routing information for one AS

<table>
<thead>
<tr>
<th>PCB</th>
<th>Core: Out: 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS E:</td>
<td>In: 1, Out: 4</td>
</tr>
<tr>
<td>AS F:</td>
<td>In: 1, Out: 3</td>
</tr>
<tr>
<td>AS B:</td>
<td>In: 1</td>
</tr>
</tbody>
</table>

**AS X:**
- In: y, Out: z
SCION Routing (Control Plane)

Routing Phases:
1. Path Exploration
2. Path Registration
3. Path Resolution

Beaconing

Path Server

PCB

Core: Out: 4

AS E:
In: 1, Out: 4

AS F:
In: 1, Out: 3

AS B: In: 1
SCION Routing (Control Plane)

Routing Phases:
(1) Path Exploration
(2) Path Registration
(3) Path Resolution

Routing Phases:
(1) Path Exploration
(2) Path Registration
(3) Path Resolution
SCION Forwarding (Data Plane)

Packet header

Forwarding along:
- Up-Segment
- Core-Segment
- Down-Segment

Segments are sequences of Hop Field (HFs).

Hop Field contain routing information of one AS.
Verification

High-level, omitting formal details
Can We Verify Scion?

- Control and data plane guarantees
- Functional correctness of actual code
  - Suitable for high-assurance business cases
  - Ensures that routers are backdoor-free
- Scion routers are simple and stateless
  - This is the key to their (feasible) verification
  - Not possible for current Internet with highly complex routers and giant code bases of millions of lines
Correctness and Security
SCION approach

Verification of the protocol at the network level

- Abstract models of network & network-wide properties
- Protocol verification guarantees that security properties hold in an adversarial environment, assuming that each SCION component behaves as specified

Verification of the components at the code level

- Code-level guarantees (e.g., secure information flow)
- Guarantees that each SCION component behaves as specified

Data Plane  ←  Initial focus  →  Router code
Network-Level Verification: Approach

- **Formal specification** of network and network-wide properties
  - Description of network topology, beaconing and path construction, ...
  - Network adversary (on and off-path)
  - Network-wide security properties

- **Formal verification**: refinement used to go from high-level models to precise assumptions on the individual components needed to ensure security properties.
  - Correctness by construction: *stepwise refinement* between (transition) systems
  - Proofs: forward simulation and invariant preservation
  - Invariants preserved under refinement

- **Tool support**: verification using Isabelle/HOL system with ETH Zurich developed theory extensions.
Scion Properties
On both control and data planes

Control planes properties: address beacons’ authenticity

• Security critical, but not in focus of this talk

Data plane properties: address how routers forward messages

• Path Authorization: Packets traverse the network only along previously authorized paths.

• Weak Detectability: An active attacker cannot hide his presence on the path.
def router():
    while (pkt.nxt()):
        pkt.process()
Concrete Attacker Model

We use a localized, colluding Dolev-Yao attacker model

Attacker controls the entire network

Attacker controls entire ASes
**System & Environment**

- Environment
  - System
- Attacker
- Network
- End hosts
- OS & Libraries
- Border Router
SCION Router Verification Overview

Model

Environment Model
- attacker, network

Router Model

Reality

Real Environment

Router Code

Protocol
- Security Properties
- satisfies

Code
- Security Properties
- refined by
  - unproven
  - justified
  - proven

Verified SCION
SCION Router Verification Overview

Model

Environment Model
attacker, network

Router Model

Reality

Real Environment

Router Code

Protocol Security Properties

satisfies

refined by

unproven

justified

proven

Code Security Properties

Verified SCION
Abstract Packet Format

The Path is the Packet

![Diagram of a packet with Past Path and Future Path]

The Path (consisting of Past and Future) contains **Hop Fields (HF)**

Example:

```
| HF1 | HF2 | HF3 | HF4 | HF5 | HF6 |
```

A Hop Field contains routing information of one AS
Refinement Overview

Communication channels | Hop Field format | Attacker
--- | --- | ---

**Idea:** strengthen attacker while increasing protection of paths.

- **Message set**
- **Neighbor ASes**
- **MAC**
- **Fields protected by MAC**
Simplified Scenario (Initially)
Packet traversal along a single up-segment

- A set of **authorized-paths** from path server is given as parameter
- Path is an up-segment. Simplify setting for now:
  - Ignore for now core- and down-segments
  - No peering or core links
  - No inter-domain communication (single ISD)
  - No changes in link status (up/down)

**Verification is still challenging enough!**
Simplified Scenario
Data Plane Model 0

Example of one Packet along a simple Path

<table>
<thead>
<tr>
<th>Past Path</th>
<th>Future Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B C</td>
<td></td>
</tr>
<tr>
<td>A B</td>
<td>C</td>
</tr>
<tr>
<td>A B C</td>
<td></td>
</tr>
<tr>
<td>A B C</td>
<td></td>
</tr>
<tr>
<td>A B C</td>
<td></td>
</tr>
</tbody>
</table>

Recv0

Up0

Up0

Send0
Data Plane Model 0

Problem: Past Path is unreliable
• Add a new component to the message: the real path
  - records the actually traversed path so far

• Not part of the system, no correspondence in implementation
  - used for property specification only
  - Corresponds to a “history variable”
Formalized Properties of Model 0

Interface with control plane: We assume a set authorized-paths that contains the paths determined by the control plane.

• **Path Authorization** Packets traverse the network only along previously authorized paths.

• **Weak Detectability** An attacker cannot hide his presence on the path. This follows from property: the real path is a suffix of the past path.
Data Plane Model 1

Hop Field format is refined:

Model 0

\[ A \rightarrow (\text{previous AS}, A, \text{next AS}) \]

Added: references to previous and next AS
Data Plane Model 1

Real Path  | Past Path  | Future Path
---|---|---

**Recv1**
- A, B, C
- (⊥, A, B)
- (A, B, C)
- (B, C, ⊥)

**Up1**
- A, B
- (⊥, A, B)
- (A, B, C)
- (B, C, ⊥)

**Up1**
- A
- (⊥, A, B)
- (A, B, C)
- (B, C, ⊥)

**Send1**
- (⊥, A, B)
- (A, B, C)
- (B, C, ⊥)
Data Plane Model 2: "Chaining" of MACs

Hop Field format is further refined by adding a MAC

- MAC at A is produced with a key(A) known only to A
- MAC includes data and MAC of subsequent Hop Field (needed for verification)

Simplified representation:

\[(\bot, A, \overrightarrow{A}, \bullet) (\overrightarrow{B}, B, \bullet) (\overrightarrow{C}, C, \bullet) (\overrightarrow{D}, D, \bullet)\]
Data Plane Model 2

Real Path → Past Path → Future Path

Recv2


Up2


Up2

A → (⊥, A, B, ●) → (A, B, C, ○) → (B, C, ⊥, ○)

Send2

(⊥, A, B, ●) → (A, B, C, ○) → (B, C, ⊥, ○)
Up-Event in Model 2

Guard

\[ \text{In} \quad \text{select} \quad \text{in} \]

Check

\[ \text{Out} \quad \text{put} \quad \text{where} \]

Action

\[ \text{Guard} \]

\[ \text{Check} \]

\[ \text{Action} \]

\[ \begin{align*}
\land \quad \bullet_1 &= \text{valid MAC using key}(A_1) \\
\land \quad \bullet_2 &= \text{valid MAC using key}(A_2) \\
\land \quad \bullet_1 &= A_2 \land \bullet_2 = A_1
\end{align*} \]
Refining Model 2

Model 2

Refinement

Model 3

Global Message Set

Inter-AS Message Sets
Up-Event in Model 3

Guard

\[ \text{In select from } \]

Check

\[ \text{Out put } \]

Action

\[ \text{Out put } \]

\[ \text{Guard} \]

In select from

Check

\[ \text{Out put } \]
SCION Router Verification Overview

Model
- Environment Model
  - attacker, network
- Router Model

Reality
- Real Environment
- Router Code

Protocol
- Security Properties

Code
- Security Properties

Verified SCION

- satisfies
- refined by
- unproven
- justified
- proven
Router Model vs. Code

Environment Model
- attacker, network

Router Model

Real Environment
- Router Code

Guard
- In
- Check
- Action

def router():
  while (pkt.next()):
    pkt.process()
  ...

Out
Main goal: prove **functional correctness**.
- Code refines the protocol.

Other desirable properties **only on code level**:
- **Safety**: Code does not raise runtime exceptions or have data races.
- **Secure information flow**: Code does not leak any information about crypto keys.
- **Liveness and deadlock freedom**

Focus on the SCION code base.
- Used libraries are given specifications, **assumed** to be correct.
- Runtime, OS, ..., are **assumed** to be correct.
Program Verification

- **Formal specification** for each method
  - Pre- and postcondition, loop invariants

- **Formal proof** that implementation satisfies specification.
  - Assuming **precondition** holds at the beginning, prove that **postcondition** holds after return (partial correctness).
  - For all possible inputs, schedules, callers, ...
  - Additional proof obligations for special properties, like progress

```
def sqrt(n):
    ...
    return result
```
Code-based Verification

- Scion in Python 3
  - ~11k LOC

- Substantial subset of Python
  - Most standard OOP features
  - e.g. inheritance, exceptions, concurrency

- Focus on router first

- Use Viper Toolchain with Python front end
Linking it all up via Input-Output Specifications (Code can be viewed as a transition system)

Guard

In  

Check

\[ \text{matches}(\text{abs}(\text{pkt}), \ldots) \land \text{check}(\ldots) \]

Action

Out  

SCION Router Verification Overview

Model

Environment Model
- attacker, network

Router Model

Reality

Real Environment

Router Code

Protocol
- Security Properties

Code
- Security Properties

satisfies

refined by

unproven
justified
proven

Verified SCION
Status

- Code verification tools built and prototyped
- First three levels of refinement completed
  - Improved understanding of protocols and properties
  - Uncovered numerous bugs and omissions
    - Revealed during modeling & formalization
    - Verified against implementation
- Next step: formally connect the two parts
Conclusions

- Internet, as designed, is insecure
- Scion architecture offers much stronger guarantees
- These can be put on a formal footing via
  refinement + code-level verification
- Long term objective: guaranteed back-door-free routers, made in Switzerland
Want to be a Scion AS?