Firenet – PhD semester project

Jingyue Zhao

Supervisors: Prof. Bryan Ford, Prof. Katerina Argyraki
Network Management
Network Management
Network Management
Motivation

• Long-term goal: a transparent and secure decentralized network management scheme for large-scale networks.

• Decisions of each administrator $\rightarrow$ direct or indirect impacts on other parts of network.

• Admins can be compromised $\rightarrow$ disaster of the entire network.
System Model

- A single small group of administrators make network policies together.
- Follower routers correspond to SDN controllers which deploy the network policies.
- Each network policy needs to be checked and approved by a threshold of admins to avoid careless or malicious actions.
System Model

- A single small group of administrators make network policies together.
- Follower routers correspond to SDN controllers which deploy the network policies.
- Each network policy needs to be checked and approved by a threshold of admins to avoid careless or malicious actions.
System Model

- A single small group of administrators make network policies together.
- Follower routers correspond to SDN controllers which deploy the network policies.
- Each network policy needs to be checked and approved by a threshold of admins to avoid careless or malicious actions.
Cothority

- Collective authority: a set of witness servers called conodes that collectively execute decentralized protocols
- Provide services: collective signing (CoSi [1]), Byzantine agreement, or generation of public-randomness

Threat Model

- **Network administrators**: make and approve network policies
  - **Malicious**: propose or approve bad policies; only a threshold of admins can be compromised by an attacker
- **Cotherity** (witness servers): check and track admins’ policy making process
  - **Honest**
- **Follower routers**: pull and deploy the network policies periodically
  - **Honest**
Design of Firenet

Step 1: admins’ approval

Key 1
Key 2
Key 3
Key 4

Network policy

Admins
Design of Firenet

Step 1: admins’ approval

Admins

Network policy
Admin signatures

Follower routers

Key 1
Key 2
Key 3
Key 4
Design of Firenet

Step 1: admins’ approval

Key 1
Key 2
Key 3
Key 4

Admins

Network policy
Config file
Admin signatures

Follower routers
Design of Firenet

Step 1: admins’ approval

Admins

Key 1

Key 2

Key 3

Key 4

Network policy
Config file
Admin signatures

Follower routers

Verify admins’ signatures & deploy the policy
Design of Firenet

Step 2: cothority’s approval check and collective signing

For now, the check is done by one server in the cothority, and we can design a protocol to distribute the workload.
Design of Firenet

Step 2: cothority’s approval check and collective signing (using CoSi [1])

Design of Firenet

Step 2: cothority’s approval check and collective signing

Admins

Network policy
Config file
Admin signatures

Cothority

Network policy
Config file
Collective signature

Follower routers

Verify one single co-signature & deploy the policy
Design of Firenet

Step 3: cothority’s appending the new policy to the chain (using Skipchain[2])

Design of Firenet

Step 3: cothority’s appending the new policy to the chain
Design of Firenet

Step 3: cothority’s appending the new policy to the chain
Design of Firenet

Step 3: cothority’s appending the new policy to the chain

Admins

Cothority

1. Block hash
   Network policy
   Config
   Collective signature
   Random ID

2. Block hash
   Network policy
   Config
   Collective signature
   Previous block hash

3. Block hash
   Network policy
   Config
   Collective signature
   Previous block hash

Follower routers
Firenet in 1 slide

Step 1: Admins’ policy proposal & approval

Step 2: Cothority’s approval check & co-signature

Step 3: Cothority’s appending new policy to the chain

Step 4: Follower routers’ downloading, verifying & deploying the latest policy periodically
Network Policy Description Language

- Based on Linux `iptables`
- One network **policy consists of several network rules**
- One policy is self-sufficient
- JSON object

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy description</td>
<td>string</td>
</tr>
<tr>
<td>Number of network rules</td>
<td>int</td>
</tr>
<tr>
<td>An array of network rules</td>
<td>Network rule 1</td>
</tr>
<tr>
<td></td>
<td>Network rule 2</td>
</tr>
<tr>
<td></td>
<td>…</td>
</tr>
<tr>
<td></td>
<td>Network rule n</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matches</td>
<td>Chain INPUT, OUTPUT, FORWARD</td>
</tr>
<tr>
<td>Protocol</td>
<td>TCP, UDP, ICMP, ALL</td>
</tr>
<tr>
<td>Source IP/network</td>
<td>x.x.x.x, x.x.x.x/x, ALL</td>
</tr>
<tr>
<td>Source ports</td>
<td>Port number(s)</td>
</tr>
<tr>
<td>Destination IP/network</td>
<td>x.x.x.x, x.x.x.x/x, ALL</td>
</tr>
<tr>
<td>Destination ports</td>
<td>Port number(s)</td>
</tr>
<tr>
<td>Action</td>
<td>ACCEPT, DROP, REJECT</td>
</tr>
</tbody>
</table>
Implementation

• Firenet is implemented in Go
• Based on the Cothority framework, using CoSi and Skipchain
• 1.3kLOC
• Main functions with APIs
  • Genesis Policy Request
  • New Policy Request
  • Get Policy Request
  • Verify Policy Request
Evaluation

Testbed: 32-core Intel Xeon CPU at 2.6 GHz with 66GB of RAM (one server of IC cluster)

Maximum 0.18 sec for 100 admins  
Maximum 20.8 sec for 128 conodes
Evaluation

Time cost component

Genesis policy CPU time component (50 admins, 128 conodes)

- ApprovalCheck: 0%
- CoSign: 20%
- CreateBlock: 80%
- Others: 0%

New policy CPU time component (50 admins, 128 conodes)

- ApprovalCheck: 0%
- CoSign: 8%
- CreateBlock: 0%
- StoreBlock: 92%
- Others: 0%
Compared to SDN

- Follower routers can be seen as SDN controllers
- Security-enhanced SDN application layer
- Easy to rollback to a previous correct network configuration

Future Work

• Performance evaluation & analysis
  • Tendency and limit
  • Bandwidth
  • Time cost vs number of policies

• Protocol improvement
  • Multiple groups of admins
  • Hierarchical network policy
Thank you!
Implementation

- **NM APP**
  - NM service
  - NM service
  - NM service

- **NM APP (leader)**
  - NM service

- **NM APP**
  - NM service

- **FR APP**
  - FR APP
  - FR APP
  - FR APP

**Admins**

**Protocols**
- Calling Cosi + Skipchain API
- Push
- Pull

**Network policy skipchain**

**Validation and Signature**
- Validate the new policy block & append it to the chain
- Return the latest policy & validate its co-signature

**Requests**
- Genesis Policy Request, New Policy Request
- Get Policy Request, Verify Policy Request