Cornerstone
Automating Remote NDN Entity Bootstrapping

Tianyuan Yu, Xinyu Ma (UCLA)
Hongcheng Xie (City University of Hong Kong)
Dirk Kutscher (The Hong Kong Univ. of Science and Technology [GZ])
Lixia Zhang (UCLA)
NDN: Named Data Networking

• **IP networking**: delivering packets to destination addresses

• **NDN networking**: fetching data by data names
  • Data with matching name can come from anywhere
  • Every data packet is cryptographically secured

• NDN needs a bootstrapping solution
Bootstrapping in a TCP/IP network

• An IP network is made of *interconnected nodes*, each identified by one (or multiple) IP address

• To connect a new IP node \( N \) into a network: \( N \) needs to go through a bootstrapping step first
  • DHCP server provides \( N \) with a set of connectivity parameters (its address, router address, etc)
  • IPv6: secure neighbor discovery based on CGA

• After bootstrapping: \( N \) can send & receive IP packets
Bootstrapping in an NDN Network

• An NDN network is made of *named entities* with *trust relations* among them.

• Applications produce and consume *semantically named, secured* data:
  • Each app process needs crypto credentials for signing, and security policies for verifying received data packets
  • Are they signed by the right key?

• To introduce a new NDN entity $N$ into a network: $N$ needs to go through a bootstrapping step first:
  • To obtaining a set of security parameters (trust anchor, certificate, security policies, etc)
  • After bootstrapping: $N$ can sign & verify data packets

• Need a “DHCP-like” NDN bootstrapping solution
A side note: does an IP network perform security bootstrapping?

• TCP/IP as designed has no security consideration
  • IP bootstrapping is all about IP connectivity
  • IPv6 secure neighbor discovery binds addresses to identities (public keys)

• Security solutions gradually introduced/patched on IP and TCP
  • IPSec, SSL/TLS, BGPSec, DNSSEC etc:
    they all need configuration of crypto credentials

• Common practice: manual or offline management of trust anchors
  • E.g., configuring Certificate Authorities into end hosts
Basic Steps in NDN Bootstrapping

- First perform mutual authentication
  - Bootstrapper verifies the authenticity of new entity
  - New entity makes sure it talks to the intended party

- Then bootstrapper installs security components into the new entity

This can be a node, a sensor, an app instance, anything that may produce and/or consume NDN data packets)
Local versus Remote Applications

Previously, several works in building local apps
  • NDN entities are local to the bootstrapper
  • Direct WiFi or Bluetooth connectivity
  • Within physical reach or the Line Of Sight
    • Smart homes, local file sharing, etc.
Local versus Remote Applications

• Recently, more works in building distributed apps → need for bootstrapping remote entities

• To bootstrap remotely located new entities
  • In an ideal NDN-native scenario
    • Remote entities already locally bootstrapped.
    • Mutual authentication by proving NDN certificate ownership

• At initial deployment:
  new entities to be bootstrapped are reached via unsecured Internet connectivity
  • Health data sharing [ICN’22]¹: remote users; the data storage server is local to the bootstrapper
  • Federated, distributed storage system (more later): all users and file servers are remote

Local Bootstrapping: Solutions Exist
Remote Bootstrapping: New Problem

Smart Home

New entity within physical vicinity, reachable through unsecured local connectivity
Authentication by QR code, vibration...

Collaborative Apps

New entity only reachable through unsecured Internet
How to authenticate?

Approach: Starting from a specific application: **Hydra**
Hydra: A Federated, Distributed Storage System over NDN

- A Decentralized Data Repository for Big Science Data

- Made of federated file servers contributed by participating organizations utilizing
  - Available storage resources at different organizations
  - NDN’s built-in anycast for data access

- Hydra Networking Operating Center (NOC) manages Hydra entities’ trust relations and read/write access
Bootstrapping Hydra Entities

- Three types of Hydra entities
  - Data Publishers
  - Storage Servers
  - Data Consumers
- They can only be reached via unsecured Internet connectivity
  - How to perform remote authentication?
Our Solution

• **Data consumers:** anyone can fetch data from Hydra
  • No need to authenticate consumers
  • Consumers need to verify data: authenticate NOC, then fetch the Hydra trust anchor and security policies during bootstrapping

• **Data producing users and storage servers:** relying on existing trust relations in today’s systems to perform remote authentications
Terminology

• **Trust Domain**
  • All Hydra entities form one Hydra trust domain, administrated by the Hydra NOC

• **Trust Anchor**
  • Hydra NOC certificate
  • Serve as the source of trust in Hydra operations

• **Certificate**
  • Issued to data producers and storage servers under their Hydra assigned names

• **Security Policies**
  • Defined by Hydra NOC, so that one knows how to verify received data (application files, coordination among data producers and storage servers)
Bootstrapping Step 1: Achieving Mutual Authentication

1. Hydra App instance authenticates Hydra NOC
   • Utilizing the authentication process of software distribution
   • Embedding Trust Anchor and Security Policies in Hydra app package
   • Requiring shared trust on software providers (e.g., GitHub, Ubuntu PPA, etc.)

2. Hydra NOC authenticates Hydra App instance
   • Servers have DNS names and TLS certificates
   • Users have their own Internet identifiers
     • Emails, identifiers from clouds
   • Membership control through identifier whitelist
Bootstrapping Step 2: Installing Security Components

• Installing Trust Anchor and initial Security policies during application installation
  • Initial security policies used for validating packet exchanges during bootstrapping

• Complete policies are fetched via Interest-Data exchange after bootstrapping

• Name Assignment

• Certificate Issuance
Name Assignment

• Each data producer and storage server is assigned a hydra name to ease the security policy definition (which makes use of semantic naming)
  • Storage server DNS name
    • bruins.cs.ucla.edu
  • Hydra name assigned to the storage server
    • /hydra/node/edu/ucla/cs/bruins

• Name assignment is automated using well defined naming conversion rules
Authentication and Naming Workflow

- **Authentication Parameters**
  - Server: DNS Name, User: email

- **Identity Proof**
  - Server: TLS Certificate
  - User: PIN Code, JSON Web Token, SAML response, etc.

New Hydra Entity

- Authentication parameters (identifiers, e.g., email)

  Please return the identity proof (e.g., PIN code, JSON Web Token)

  Here's the proof

  Here's the name assignment

NOC

1. Initiating authentication

2. Identity Verified!

3. Identifier in the list?

4. Converting identifier into Hydra name
Certificate Issuance

• The requester has a name assignment
• The issuer
  • Requires the requester proving its name possession
  • Certifies the requester’s name
• Today’s Let’s Encrypt
  • Asks the requester for Proof of Possession
    • By generating a DNS record, serving an HTTP object, ...
    • Trusting the DNS and routing system
• Hydra’s approach
  • Proof of Possession = NOC signed name assignment
Proof of Possession

Hydra App proving it is a legit owner for a name
  • Proving public key ownership == Proving the name possession
  • NOC signed Data packet, in certificate format

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Assigned Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: /authenticate/hydra/node/edu/ucla/cs/bruins/KEY/123/auth-signer/v=1</td>
<td></td>
</tr>
<tr>
<td>Content: Server’s Public Key</td>
<td></td>
</tr>
<tr>
<td>Signature</td>
<td></td>
</tr>
<tr>
<td>ValidityPeriod: 20230228T0308 – 20230301T0308</td>
<td></td>
</tr>
<tr>
<td>KeyLocator: /hydra/auth-signer/KEY/223/anchor/v=1</td>
<td></td>
</tr>
<tr>
<td>SignatureValue</td>
<td></td>
</tr>
</tbody>
</table>
Proof of Concept Implementation

https://github.com/tianyuan129/ndn-bootstrap
Configuring Hydra NOC

identity_config: /hydra
auth_config:
  user:
    whitelist: [alice@cs.ucla.edu, bob@ucla.edu]
  server:
    trust_anchors: /etc/ssl/certs
    whitelist: [bruins.cs.ucla.edu, suns.cs.ucla.edu]
validity_period: 3600s
Takeaway 1:
IP vs. NDN: Commonality in Bootstrapping

- Identify the necessary knowledge to be configured into bootstrapper
- Automate the rest to minimize human errors

<table>
<thead>
<tr>
<th>DHCP</th>
<th>NDN Bootstrapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Allocated IP address block to use</td>
<td>• Namespace to use</td>
</tr>
<tr>
<td>• Policy</td>
<td>• Naming conventions</td>
</tr>
<tr>
<td>• e.g. setting MAC address restrictions</td>
<td>• Policy</td>
</tr>
<tr>
<td>• Lease durations</td>
<td>• Authentication rules</td>
</tr>
<tr>
<td>• Other DHCP options</td>
<td>• e.g. what/which entities to be admitted</td>
</tr>
<tr>
<td></td>
<td>• Certificate lifetime</td>
</tr>
</tbody>
</table>
Takeaway 2: Specifics in NDN Bootstrapping

• Before NDN gets widely deployed
  • Utilizing existing existing identifiers to authenticate remote entities
    • e.g. email address, DNS name
  • Utilizing the existing trust relations to carry out mutual authenticate between remote entities
    • e.g. trust the authenticity of files stored on git system

• Utilizing naming convention
  • Automates name assignment
  • Facilitates security policies definition