Secure Off-Path Replication in Content-Centric Networks

Marc Mosko
PARC

Christopher A. Wood
UC Irvine

NGNI: Future Internet and Next-Generation Networking Architectures
ICC 2017, Paris, France
May 22, 2017
Agenda

• Background, Problem Statement, Related Approaches
• CCN Overview
• SCR Design
• Analysis
• Conclusion
Background

• CCN and NDN are two prominent Information Centric Networking (ICN) technologies.

• A consumer asks for data by a name

• The request is routed by name to the producer

• The data may be cached anywhere and retrieved by anyone with the name (or possibly even discovered by a name prefix).

• Access control via encryption
Problem Statement

ICN blind caching is dangerous

• Forwarders do not enforce access control and must allow anyone to access data if given the right name

• Producers have no knowledge about where content is cached

• Producers compete for cache space and may starve others

Off-path caching is not practical without significant protocol or storage requirements at intermediate forwarders
Proposed Approach

• Build a semi-trusted caching system in CCN

• Producers store content on known caches

• Consumers request pointers and security material from producer

• A consumer securely fetches data from one or more caches (in parallel)

• Protects against off-path adversary guessing names, fetching content
IPBC
(HTTPS Blind Caching)

• HTTPS-based proposal solving similar problem

• Servers publish static content in CDN caches

• Clients request index pages over HTTPS from source

• Servers specify the decryption key(s), location, and hash digest of desired content

• Work on our approach in CCN was concluded in Jan 2015, over a year before publication of draft-thomson-http-bc-00.
CCN Overview

• All data cryptographically bound to a name

• Producers transfer data to consumers upon explicit request

  • **Consumers** of data issue *interests* for data **carrying the name**

  • An *interest* may include a **cryptographic hash** of the expected response, which could be verified anywhere.

  • **Producers** reply to requests with the *named data responses*

  • **Forwarders** relay requests and responses
For Warder Behavior

Caching

Interest
Data
Content Store

Pending Interest Table (PIT)
add incoming interface

FIB
forward

drop or NACK

Aggregation

Forwarding

Downstream
forward
Content Store

Data

Upstream

Pending Interest Table (PIT)
discard Data

\( \times \) lookup miss
\( \checkmark \) lookup hit
CCN Components

• Interest: a request carrying the name of some data

• Content Object: a packet carrying the data (and name) corresponding to an interest

• FLIC: a packet carrying “pointers” (names) to other content objects (a manifest)

• CCNxKE: Name-based TLS 1.3-like key exchange

• IBAC: Interest-based access control
FLIC

https://www.ietf.org/id/draft-tschiudin-icnrg-flic-03.txt

Mosko & Woods, “Secure off-path caching in CCN”
CCNxKE Overview

• Protocol used to set up “sessions” between a consumer and entity servicing a namespace

• Based on TLS and related protocols

IBAC Overview

• Consumers use name encryption to restrict access to content
• Producers can decrypt names to identify the right content response
• No handshake is needed (if keys are established out of band)

Proposed Approach

• Secure Content Replication (SCR)
  • Producers publish encrypted static content in trusted replicas
  • Consumers fetch FLIC roots for static content using IBAC or CCNxKE session
  • Consumers resolve the FLIC tree from the replicas (in parallel)
SCR Process

1. Name $N$, data $D_N$, set of Links $\{L_i\}$ to replicas $R_i$

2. Encrypt data $D_N \rightarrow (C_N, \text{security material})$

3. Build FLIC transport manifest over encrypted data $\rightarrow T_N$

4. Create a signed Root Manifest

   $\text{ROM}(N) = (N, \{L_i\}, H(T_N), \text{security material})$
SCR Pictorially

Root Manifest

Transport Manifest

Encrypted Data

\begin{align*}
\{ \text{ROM}(N) \} \quad & (N, \{L_i\}, H(T_N), \ldots) \\
\{ \text{T}(N) \} \quad & R_j \in \{L_i\}, H(T_N) \\
\{ \text{FLIC Node} \} \quad & R_j, H(\text{node}) \\
\{ \text{Data Node} \} \quad & R_j, H(\text{node}) \\
\end{align*}
SCR Properties

• Root manifest transferred over encrypted channel to protect \{L\} and \(H(T_N)\) and to distribute consumer-specific keys

• Content stored on caches uses hash-based naming (e.g. 256-bit pseudo-random strings) and is group encrypted

• a consumer/adversary cannot (with vanishing probability) guess the name of content they have not already asked for

• Provenance comes from signed ROM and hash chains, consumer can verify data at every step
<table>
<thead>
<tr>
<th></th>
<th>IBAC</th>
<th>Session-Based</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C - P</strong></td>
<td>Pros: One RTT to obtain replica information, replica information may be cached</td>
<td>Pros: Efficient response processing at Producer, MoveToken support for replica resumption</td>
</tr>
<tr>
<td></td>
<td>Cons: Computational bottleneck for a single Producer</td>
<td>Cons: Session state storage, Multiple RTTs to fetch data</td>
</tr>
<tr>
<td><strong>C - R</strong></td>
<td>Pros: Minimal number of packets to fetch</td>
<td>Pros: Efficient data transfer once session is bootstrapped</td>
</tr>
<tr>
<td></td>
<td>Cons: Larger computational bottleneck</td>
<td>Cons: Sessions are pinned to specific replicas</td>
</tr>
</tbody>
</table>
Analysis
Analysis
Conclusion

• SCR compares well with IPBC

• SCR offers more flexibility in terms of the desired AC-enforcement mechanism than IPBC

• Either IBAC or CCNxKE sessions can be used

• Results may be verified at each step

• Content striping retrieval from multiple replicas

• Consumer-based replica selection