A Verify & Pushback Solution to Content Poisoning

Steve DiBenedetto
Christos Papadopoulos
Colorado State University
NDN Background

- Route on content names
- “Breadcrumbs” & de-duplication of requests
- Cache retrieved data
NDN Packets

**Interest Packet**
- **Content Name:** Identifies the data I want to receive
- **Selector:** identifier publisher, etc
- **Nonce**

**Data Packet**
- **Content Name:** Identifies the data in this packet
- **Signature:** Required for all packets
- **Data**
Poisoning Prevents Content Retrieval

- Interests forwarded to sources serving bad objects
- Cached bad copies block future requests
- Must fix BOTH before user abandons the content/service (<4s)
Bad Content Objects

• Malicious, misconfigured, and corrupted objects are “bad”
  
  – Fail to verify
    • Untrusted key
    • Unauthorized key
    • Wrong signature
  
  – OR cannot be verified
    • No key locator
    • Unreachable key
    • Excessively long key chains
  
• Trust decisions make “bad” subjective
Current (Non-)Solutions

• In-network verification is a complete solution, but does not scale
  – Computationally expensive and may require key fetch(es)

• Secure routing is oblivious to most bad content
  – Can retrieve misconfigured and corrupted content from paths to “legitimate” sources
    • Compromised routers are a rare problem

• Common questions:
  – Who defines legitimate?
  – What happens when nodes disagree?
Trust Partitions

- Trust disagreements prevent content retrieval
- Network can route around disagreements like any other failure
Main Ideas

• Network is responsible for delivering legitimate content

• In network detection is burdensome
  – Consumers can detect, but have limited power to act

• Content from anywhere – no preferred content origin
Assumptions

• Content publisher’s key is not compromised

• There is a feasible path to desired content

• Rough consensus on which Content Objects are bad
Proposal: Report-Driven Pushback

- Leverage consumers for bad object detection
  - Retrieve keys and verify objects
- Upstream confirms problem and propagates warning
- Change forwarding strategy to avoid bad content
Report Validation

• All network nodes/routers validate reports

• Trust centered on content, not reporter
  – Verifying objects produces a locally correct answer
  – Nodes may reach conflicting answers

• Only process reports from well-behaved (whitelisted) downstreams
Implementation

• Reserve a global namespace for reporting
  – /report/<Upstream>/<Downstream>: unique FIB entry for each directly connected downstream
  – /report: null route to filter unwanted reports

• Reports are Interest in the form of:
  – /report/<Upstream>/<Downstream>/name/digest

• Record each Content Object’s arriving face to direct report to correct upstream
Example: Reporting

FIB:
/  
/report £ null  
/report/2/1  
...

/report/2/1/youtube/video/<digest>

/report/3/2/youtube/video/<digest>
Reports Follow the Content Path

Already processed

Routers should only see reports for content it relayed
Handling Bad Reports

FIB:

/  
/report \ null  
/report/3/1  
/report/3/2  
...  

/report/3/2/youtube/video/<digest>

- Rough trust consensus means most disagreements are malicious
- Suppression of legitimate reports across namespaces implies divergent trust views

Does my copy verify?

Temporarily suppress reports if object verifies
Key Retrieval

- Routers responsible for bad content may not be on the key retrieval path
- Reporting downstream must have keys if report is legitimate
- Push the whole keychain with the report
Attacking Key Retrieval

- Key Retrieval depends on an untrusted object’s locator field
- Attacker will generate distinct key chains per object
  - ... and delay serving as long as possible
What is the Right Key?

Roots pre-installed and likely have special namespaces:
/verisign-root/key vs. /verisign/key

```
    root
     ↓
   /colostate
      ↓
   /colostate/key
       ↓
 /colostate/netsec
       ↓
 /colostate/netsec/key
        ↓
 /colostate/netsec/steve
        ↓
 /colostate/netsec/steve/key
       ↑
```

Required keys are obvious from the content’s name
Adapting Forwarding Strategy

- Need to avoid bad upstream copies until problem is fixed
- Strategies trade-off failover speed and success rate
  - Immediate failover is risky if poisoning is widespread
  - Delaying failover to probe upstreams is less risky, but may require key retrievals
Current Progress

• Prototype in development
  – Experimenting with forwarding strategies

• Evaluate on PlanetLab
  – Measure retrieval time under poisoning
  – How usable is a VoNDN?
Summary

• Content poisoning is a multi-part problem
  – Interests forwarded towards sources serving bad objects
  – Cached bad objects block Interests
  – Not just malicious objects: misconfigs and corruption, too

• Established key principles for an NDN content poisoning solution
  – Network responsible for delivering legitimate content
  – Consumers likely to verify anyways, take advantage of it
  – No preferred content origin

• Propose a verify & pushback solution for content poisoning in NDN
  – Consumers detect and can notify network
  – Network elements independently verify attack and propagate warning
  – Adapt forwarding strategy to avoid bad objects
Future Work

• Evaluate impact of trust disagreements

• Make trust decision results available
  – Enable network to proactively form consensus zones
Thanks!

Questions/Comments?

dibenede@cs.colostate.edu