Mitigating Poisoned Content with Forwarding Strategy

Stephanie DiBenedetto*

Christos Papadopoulos

Colorado State University

(*now at Google)
NDN: Poisoning Prevents Content Retrieval

- Interests forwarded to sources serving bad objects
- Cached bad copies block future requests, but the problem persists until forwarding changes
- Content poisoning exists in the absence of caches
IP Analogy: Routing Hijacks

• A malicious/misconfigured route has been injected into the system

• Same pattern as ICN content poisoning
  – Packets forwarded to bad source (or dropped)
  – No in-network caching

• Solutions focus on securing a path to a blessed origin
  – Reminder: IP security is pipe oriented
IP Solutions Don’t Fit NDN

- NDN content poisoning exists at the network layer
  - Applications cannot communicate out of band to resolve the problem
  - ALL communication is vulnerable to content poisoning

- Secure routing is poor fit for NDN’s content-centric security model
  - Don’t want to be limited to a blessed origin
    - Destroys the notion of content from anywhere
  - Compromised routers (though rare) are still an issue
Back to Basics

• Data packets have built-in provenance – why not use it?

• Suppose every node verified every packet
  – First contact node stops poisoned content from propagating
  – Hopefully, the downstream network will timeout and try something else

• Unfortunately, this doesn’t scale
  – Verification is computationally expensive: \( O(10,000 \text{ ops/s}) \)
  – Requires fetching keys: \( O(10 \text{ fetches}) @ O(100 \text{ ms}) \)
Verify Everything: Going Deeper

- **Reminder:** verification only provides detection

- “Verify everything” only works as well as the network’s reaction
  - We’re still stuck if the downstream network doesn’t try another path or chooses poorly

- Can we explicitly trigger a reaction?
Premise

• Content from anywhere – no preferred content origin

• The network has a responsibility to deliver legitimate content

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

• There is a feasible path to desired content

• No expired or compromised keys
  – Ongoing work in the greater NDN team
Proposal: Report-Driven Pushback

- Leverage consumers for bad object detection
  – Retrieve keys and verify objects
- Upstream confirms problem and propagates warning
- Forwarding strategy adapts to avoid bad content
Advantages

• Our approach features detection AND reaction
  • We are explicitly informing strategy of a problem and use available info to explore alternatives

• Verification is centered on content, not the reporter
  • Don’t care where content came from – just whether it’s allowed to have its name
  • Minimal effort: process started by those verifying anyways and network just spot checks

– Network is enforcing ownership/reachability
  • We are removing subjectivity; content either verifies or doesn’t
Implementation

• Record extra information for Data packet
  – Arrival face to direct reports upstream
  – Content name to guard against “no cache lifetime” attacks

• Create unique, scoped, reporting namespace for each directly connected downstream
  – /localhop/<Upstream>/report/<Downstream>

• Report is an Interest with a name of:
  – /localhop/<Upstream>/report/<Downstream>/<Bad Data & Keys>

• Only process reports from well-behaved (whitelisted) downstreams
  – Could just as easily use proof of work, etc.
  – Goal is simply to keep node from becoming overwhelmed by bogus reports
Example: Reporting

FIB:
/localhost/2/report → drop
/localhost/2/report/1
...

1

2

3

4

/localhost/2/report/1/<Segment 0 & Keys>

/localhost/3/report/2/<Segment 0 & Keys>

YouTube
Reports Follow the Content Path

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

FIB:
/  
/localhop/3/report → drop  
/localhop/3/report/1  
/localhop/3/report/2  
...  

/localhop/3/report/2/<Segment 0 & Keys>

- Malicious consumer is delisted after a single false report
- Relisting is subject to local policy
  - Goal is to avoid becoming overwhelmed by false reports

Have I seen this packet?

Temporarily suppress reports if object verifies
Evasion Strategies

- Need to avoid bad upstream copies until problem is fixed
- Create composable poisoning adaptation modules called *evasion strategies*
- 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 can’t forward to namespace while probing
Evasion Strategy Implementation

• Forwarding strategies leverage next hop costs to make decisions
  – Costs initialized set statically or by routing protocol

• Poisoned content is just another kind of feedback

• Evasion strategy is a pre-processor before forwarding strategy
  – Make next hop cost least preferred when problem detected
  – Existing forwarding strategy operates as normal
Evaluation Overview

• Simulate evasion strategies on top of single best route forwarding strategy using ndnSIM
  – Immediate Failover (IF)
  – Probe First (PF)
• Generate scenarios using a real world ISP topology (52 nodes)
  – Randomly place good/bad producers and consumers
  – 10,000 placements for varying numbers of each role
  – Constrain to scenarios that have a bad producer-free path to the good producer (no cut)
• Each “round” consists of a fetch attempt and report if poisoned content returned
Rounds to Completion

- Scenario: 1 good producer and consumer against N bad producers
  - Total nodes: 52
- Even with 20 attackers, both solutions typically converge in ≤ 10 rounds
  - PF does so in 90% of scenarios
- Results show promise for automatically defeating prefix hijacks
More Consumers == Faster Convergence

- Duplicate reports are ignored, so this is the effect of reports reaching different parts of the network
- Allows overall state of network to improve faster
- Implies retrieval for globally popular content will improve faster than only locally popular content
Remaining Poisoned Nodes

- Imm. Failover (w/ Best Route) clears poisoned content and tries again
- PF’s probing results in collateral poisoning because bad results are not reported
  - Could report, but no obvious halting condition if bad results continue
  - Future work can experiment with simple heuristics (back off, fixed number of tries, etc.) to improve the situation
Summary

- Content poisoning is a multi-part problem
  - Interests forwarded towards sources serving bad Data
  - Cached bad objects block Interests
- 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
  - Adaptation is essentially the standard strategy feedback cycle
Name Ownership, Not Trust

• Past work avoids in-network verification to bypass trust issues
  – NDN does not impose a trust model on applications
  – Unclear how to make network aware of/trust application-specific schemes

• The network cares about reachability, not application trust semantics
  – Reachability in ICN: trying to retrieve authentic content that could be anywhere
  – Content is authentic if it was generated by the namespace’s owner
Obtaining Ownership Info

- Rely on external system to looking up namespace ownership (keys)
  - e.g. NDNS: effectively DNSSEC-secured DNS for NDN

- Improves key retrieval by making it top-down instead of bottom-up
  - Always start from a respected anchor
  - Short circuit on verification failure