Mitigating Poisoned Content with Forwarding Strategy

Stephanie DiBenedetto, Christos Papadopoulos
Department of Computer Science Colorado State University
Email: $\{Stephanie.DiBenedetto, christos\}$@colostate.edu

***Abstract*—Content poisoning attacks are a significant problem in Information Centric Networks (ICN), such as Named Data Networking. In a content poisoning attack, an attacker injects bogus content into the network with a legitimate name. While users will reject the content because of signature mismatch, the network is largely unaware of the problem due to the computational burden of on the fly packet verification. Thus, subsequent requests may continued to be answered by bogus content and constitute a denial of service attack.While NDN could resist poisoned content by putting restrictions on prefix advertisement, the latter interferes with the "content from anywhere" principle, which we consider to be a great advantage of NDN.**

This work explores the problem of content poisoning in depth and surveys the state of the art in mitigation mechanisms. We then present a novel system for detecting, reporting, and avoiding poisoned content that leverages the verification work that users must do anyways. We also propose the use of *evasion strategies*: pre-processor modules that assist forwarding strategy in avoiding bad content sources. We evaluate two evasion strategies, *Immediate Failover* and *Probe First*, that capture the spectrum of possible solutions to avoiding bad content.

I. Introduction

Content poisoning attacks are a significant problem for information-centric networks (ICNs). In these attacks, malicious end hosts or even routers respond to requests with bogus content, potentially spreading such content around the network. Users may not necessarily accept such content. Named Data Networking (NDN), for example, is an ICN architecture that requires publicly verifiable signatures on every Data packet so that any node can detect malicious/tampered content and refuse to accept it. When bad content spreads, however, it makes it hard for users to locate valid content, leading to a degradation or denial of service. Note that content poisoning is substantially different from *cache poisoning*; content poisoning would still exist if there were no in-network caches since malicious providers can still serve bad content.

One way to eliminate poisoned content from a NDN network is for all routers to verify each Data packet as it passes through. This, however, places a significant burden on the routers since verification is not a lightweight process. While fast, cost-effective in-network verification may be possible in the future, the current working assumption in the ICN community is that verifying every packet is not practical. The problem is not just computational power; verification also requires key retrieval, which would be prohibitively time consuming if done on-demand. These costs become less justifiable if content poisoning turns out to be a rare event.

One might assume that restricting content advertisements through some sort of secure routing advertisements (i.e., verified routing announcements and propagation analogous to secure BGP) would be sufficient to eliminate the problem. We believe that traditional secure routing is too restrictive for ICNs, such as NDN, because secured data, not blessed hosts and the paths to them, is the foundation of security. Furthermore, content poisoning cannot be completely eliminated by secure routing schemes; bad content may still be injected by malicious nodes along the secure path [13]. A key source of ICN/NDN efficiency is that legitimate content can be served from *anywhere*, thus supporting new opportunities and applications.

The state of the art in poisoning mitigation focuses on detecting and removing bad Data. However, this does not avoid retrieving bad Data in future requests. In this paper, we depart from the current understanding of the problem of content poisoning and cast it as a forwarding problem: routers are unknowingly forwarding requests to bad content sources.

Poisoned content presents two problems: (1) cached bad content inhibits retrieval of good content, and (2) the network unwittingly prefers a route to a bad source. To recover, the network must remove bad Data packets from caches and forward Interests towards a different content source. However, routers must first detect bad Data packets before they are able to take any action. In comparison, end hosts *must* verify every Data packet as part of normal operation, but lack the authority to influence forwarding decisions. Thus, there is an opportunity to capitalize on work that must happen anyways. The network can be warned of a specific problem without needing to verify every packet on demand. Such warnings are possible, despite coming from untrusted devices, *because the bad Data packet itself is evidence of the problem.*

NDN already has a mechanism to deal with multiple paths: forwarding strategies. Briefly, strategies are (potentially) stateful, per prefix, “programs” that decide which next hop(s) should be selected for Interest forwarding. For example, a simple policy might be to forward Interests to next hops that have historically provided the fastest response. Note that while the forwarding strategy layer can perform its own measurements, it also allows for external input that may influence the forwarding decisions (e.g. routing setting its metrics/costs).

We explore a hop-by-hop poisoned content warning system that is bootstrapped by network/end host cooperation. Upon detecting poisoned content or receiving a warning, each node propagates the message upstream and treats the event as feedback – just another metric to guide the selection of forwarding paths, thus capitalizing on NDN's ability to retrieve content from multiple locations. In other words, we treat the path to poisoned content as a bad forwarding choice and use NDN's forwarding strategy capabilities to select a better path. We believe these semantics are consistent with the original intent of forwarding strategy to improve performance and robustness.

We present two composable, forwarding strategy modules, termed *evasion strategies* that take advantage of bad Data feedback and try to find better paths. We simulate their behavior on a real world router topology. Both evasion strategies automatically find paths to legitimate content and, by extension, defeat a prefix hijack *without operator intervention*. Furthermore, both do so in 10 or fewer iterations in approximately 70-90% of scenarios, *despite malicious nodes representing nearly half of the network*.

II. A Survey of NDN Poisoned Content Mitigation

Content poisoning mitigation schemes have recently become an active research topic. One of the simplest concepts is to use the Interest's exclusion filter to specify Data packets that cannot satisfy the Interest. For example, an Interest for name `/A` excluding `B` prevents any Data named `/A/B/*` from matching. However, relying on exclusion poses several problems. First, exclusion knowledge is limited to each consumer. Other consumers must repeat the request/exclude cycle to identify the desired Data. Second, the filter is intended to specify content ranges to eliminate (e.g. versions before 100) rather than enumerating many discrete items. Attackers can indefinitely produce malicious Data, thus leading to an unscalable exclude filter. Exclusion also cannot help consumers requesting Data by a less specific name prefix. Suppose legitimate content is named `/A/B/D` and malicious content is named `/A/B/C`. If a consumer requests */A*, both the legitimate and malicious content are valid matches. The consumer **cannot** exclude name component `B` because ignoring all names prefixed `/A/B/*` would also exclude the legitimate content `/A/B/D`. Thus, the consumer is unable to retrieve legitimate content because exclusion cannot express any finer granularity.

Approaches that rely on exclusion request patterns, such as Ghali *et al.*[8], must ensure that they do not mistakenly flag legitimate exclusion/request patterns as an indication of bad Data. Exclusion is also an important part of the NDN discovery exchanges like latest version discovery. Latest version discovery has historically required the consumer to exclude the latest Data version to ensure that there is no later version. The approach described in [8] will erroneously flag this scenario because the follow up Interest checking for any newer content may timeout.

Another poisoning mitigation approach is for Interests to specify the Data's expected publisher. The current NDN packet specification [1] supports a `Key Locator` field in Interests and Data that is either the NDN name or digest of its publisher's key. However, neither of these options is effective because an attacker can simply use the correct key information in malicious Data packets.

Kim *et al.* [10] avoids verifying Data packets until they are served from a node's Content Store. However, the underlying forwarding problem is not addressed and would result in the poisoned content being re-requested.

Ghali *et al.* [9] propose the "Interest Keybinding Rule" (IKB) that requires all Interests to specify the publisher via key digest and for Data packets to include the associated public key (rather than just a locator field). Routers then verify each Data packet with the attached key and ensure the key matches what is specified by the requesting Interest. This avoids burdening routers with multiple verifications per Data and, perhaps more importantly, having to understand application specific trust semantics and key revocations. However, each router must verify every Data packet on the fly, which we previously dismissed as impractical. Furthermore, the end result is a mandatory verification with little meaning; routers have no understanding of whether the Data is legitimate.

The more fundamental problem with using exclusion and IKB is that they do not correct the underlying forwarding issue; interests will continue to be sent towards bad sources indefinitely. We observe that there is a naming problem at the core of the issue: *these approaches imply that there can be multiple, legitimate, Data packets with the same name*[1]. NDN content names are unique before considering the implicit digest [16] and are the primary means for differentiating content.

In this light, name collisions are attacks on Interest/Data matching. *There can be only one legitimate packet with a given name*. A content poisoning solution must determine *which* packet it is in order to restore forwarding towards a good source. The network cannot allow two distinct packets to have the same name because it can lead to starvation. Ghali *et al* [9] assumes that routers are also using alternate paths that deliver legitimate content. However, this depends on the specific forwarding strategy and cannot be guaranteed (e.g. a single best route strategy exclusively uses a malicious content source). The authors do not propose a mechanism for detecting this situation or how to recover from it. Thus, while IKB can prevent users from unknowingly retrieving poisoned content, it cannot guarantee the reachability of legitimate content. This is because NDN forwarding (i.e. FIB entry and next hop selection) exclusively uses names. Selectors, such as the Key Locator, are only considered during the Interest-Data matching process, not forwarding.

The NDN architecture provides a mechanism for Interests uniquely specify the desired content. Every NDN Data packet can be represented by a unique hash digest. Consequently, Interests that specify the desired Data by their exact names with digests are immune to content poisoning attacks. However, the problem is shifted to discovering the exact name. Gasti et al. [6] links each Data packet to its predecessor by placing its digest in the predecessor's payload. Unfortunately, this shifts the problem to discovery of the exact name of the initial Data packet and prevents efficient content retrieval since the consumer will not be able to pipeline multiple Interests.

Baugher et al. [5] and Kurihara et al. [11] define content catalogs and manifests to improve content publication and retrieval. Manifests define collections data by name and digest.

[1]Before considering their implicit digest.

Consequently, publishers may be able to limit themselves to signing a limited number of manifests rather than each individual Data packet. Similarly, content retrieval can then exactly specify Data, thus avoiding content poisoning, and perform fewer signature verifications.

Gasti et al. [6] also sketch a wide range of approaches for distributing the cost of verifying Data in network and, alternatively, reporting problems from consumers. However, the proposed approaches are exclusively probabilistic and do not describe how forwarding should adapt to bad content. We propose a deterministic consumer reporting system and forwarding strategies for discovering legitimate content.

III. Enabling In-Network Verification

Mandatory, publicly verifiable, signatures are the core of NDN's Data packet security. We attempt to use Data signatures as the foundation for a secure forwarding system. Interests and their poisoned Data responses form a feedback cycle that can be leveraged by nodes to steer forwarding strategy away from bad content sources. However, this assumes that arbitrary nodes can conclude that a particular Data packet is bad in the first place. In-network verification is a contentious topic because NDN does not impose a trust model on applications. However, in-network verification is a powerful mechanism for defending against poisoning attacks if we can separate its semantics from application-level trust.

We argue that *the network has no interest or role in deciding if Alice trusts Data published by Bob – only whether Bob is authorized to publish said Data under the used namespace.* Alice may not know whether or not she trusts Bob until she retrieves his key and applies the appropriate trust model. However, Alice must know that she intends to send an Interest to Bob's namespace (though, she may not know if Bob is its owner). The network is strictly limited to delivering Interests to appropriate destinations. Thus, the network should only be concerned with whether Bob is authorized to publish in a particular namespace. This is essentially the IP notion of reachability expanded to ICN. Content may come from anywhere, so control shifts to Data having a particular name.

Data packet names are unique before considering the implicit digest component. This implies there will be some form of coordinated namespace assignment. Assignment authority signing keys would therefore be well known and pre-installed on every NDN node. Systems like NDNS [2] provide the necessary lookup mechanism for determining ownership. Any node can leverage such a directory to determine the authorized key(s) for a given namespace. Past work has similarly used DNSSEC for securing the IP routing system [7]. Tamper evident revocation loggers are also in development to further aid the verification process [15].

The directory service model does not mean that applications must directly sign their Data with one of these keys. Doing so would effectively limit all applications to a hierarchical trust model. Instead, ownership keys registered with a directory service act as key signing keys that endorse other, application-specific trust model keys. This effectively glues the application's trust model to a hierarchy. More concretely, Bob may register some keys for any namespaces he owns in a directory service. These keys are signed by whomever delegated the namespaces to Bob. Bob can generate, use, and discard application-specific keys as he sees fit, so long as he endorses them with the appropriate directory-registered key. Note, however, that Alice and Bob's applications can be completely unaware of the directory service hierarchy's endorsement or involvement; they need only to deal in the trust model semantics they have been configured with. Likewise, the directory service model minimally, if at all, detracts from anonymity. For example, an anonymous message board operator can freely delegate ownership of subnamespaces to users without tracking.

IV. System Design

NDN requires every Data packet to be signed, but retrieving keys and verifying each is prohibitively expensive. However, a verification-based approach is desirable because it is content-centric. Just as signatures support retrieving Data from any source, bad signatures allow any verifying node to securely inform others of the problem. Signatures make each bad packet their own evidence, thus allowing each node to independently reach a decision based solely on the properties of the packet. This avoids problems such as trusting other nodes based on some potentially fragile authority structure, or the lack thereof. Consequently, any node with the ability to verify can detect bad Data and warn others with minimal, if any, pre-arrangement between nodes. However, Interests will continue to be forwarded towards and satisfied by malicious Data packets so long as the network prefers a path to a bad content source. Network elements must be able to explore alternative forwarding options to restore legitimate Data retrieval.

Consumer applications detect poisoned content by verifying retrieved Data packets. Applications are expected to be configured with trust anchors and any other information needed to verify Data. For ease of exposition we assume that applications have any keys required to verify a particular Data packet. The actual verification is assumed to be done via system libraries/NDN protocol stack on behalf of the application (i.e. appropriately configured). In our proposed system the stack automatically generates a special Interest called a *report* to inform the upstream network about verification-related problems. This includes problems such as unreachable, non-existent, or misconfigured keys in addition to outright signature verification failure.[2]

Report Interest names have the following form:
`/localhop/<Upstream-ID>/report/<Self-ID>/<Bad-Data>/<Keys>`
The report is scoped to be valid only between the report issuer and receiver via the `localhop` reserved namespace and `Upstream-ID` identifies the specific upstream next hop. `Upstream-ID` can be any sufficiently unique identifier, such as an identity key's digest. Reports are only sent to next hop upstream that returned the poisoned content. We anticipate NDN nodes retaining packet arrival face information and a

[2]We assume packet checksums are handled by layer 2.5 instead of relying on the signature. Similar topics are under active discussion.

mapping to the associated `Upstream-ID`. Reports also identifies the sender for accountability (`Self-ID`) via a shared secret generated by the upstream node. The remaining name components carry the bad Data packet and its signing keys as (Data packets). This uniquely identifies the offending packet and provides the upstream node with everything necessary to independently verify the report.

The report Interest is heavyweight; it carries the full bad Data packet and its keys. We assume NDN uses hop-by-hop fragmentation [4], thus mitigating problems due to large packet size. While it is more idiomatic to pull Data, doing so can cause problems in a content poisoning scenario; any pull can potentially retrieve bad Data.

Nodes run a management process to handle reports. The process registers one prefix per adjacent node:
`/localhop/<Upstream-ID>/report/<Self-ID>`
Upon receiving a report, the process first checks that the reported Data has recently been seen. The Content Store does not provide a complete list. Nodes should also maintain a supplemental list of recently seen Data names (i.e. including SHA-256 digest) and arrival faces to defend against malicious Data with no freshness period.This information, or extensions of it, may also be useful to CS replacement policies and forwarding strategies.

Next, if the reported Data has recently been seen, the management process tries to confirm the reporter's claim by checking the provided keychain against its own configured keys. If the report verifies, the node management process generates a new report for its own next hop upstream. The bad Data's name and arrival face is then given to the active forwarding strategy for each prefix matching the packet.

However, if the Data packet has not recently been seen, its signature successfully verifies, or the provided keychain is incorrect, then the report is viewed as an attack. The upstream responds to attacks by temporarily removing the reporter's unique prefix, thereby dropping successive reports. De-listing time is at the discretion of the network operator. Restoring a misbehaving reporter's entry too early costs the verifying node one set of verifications per de-listing and a new secret (`Self-ID`) must be shared.

A. Discovering Alternative Content Sources

Reporting bad Data to the network allows upstream nodes to remove cached copies. However, consumers will still be unable to retrieve legitimate content because the network is forwarding Interests to some bad source – the network must explore alternative content sources. NDN uses forwarding strategies to quickly adapt to events such as reachability and performance changes. We believe strategies should also react to poisoning attacks.

We developed a set of composable modules called *evasion strategies*. Normally, forwarding strategies choose a subset of next hops for Interest forwarding based on some algorithm (e.g. lowest cost). Evasion strategies are pre-processors that reduce the forwarding preference of next hop choices that lead to bad content. Any forwarding strategy can then use the updated preferences for the actual Interest forwarding decision. The current NDN forwarder implementation, NFD, distills everything that could affect forwarding decisions to a single integer cost where lowest cost wins. Therefore, evasion strategies would only need to understand "higher cost is worse" instead of the details of a specific forwarding strategy.

Forwarding strategy is a local decision. There is no strategy-to-strategy communication between adjacent nodes. At present, each node acts on its own according to whatever metrics are available/collectible. However, there may be some level of cooperation within an administrative domain (e.g. via SDN) in the future.

We developed two evasion strategies representing different extremes:

- **Immediate Failover:** Make the next hop that returned bad Data the least preferred option for future Interests.
- **Probe First:** Stop forwarding Interests for the namespace(s) under attack and probe all next hops. Verify the returned Data packets and resume forwarding to next hops upon successful verification.

Immediate Failover is an optimistic strategy that assumes the poisoning attack has a limited impact on the network (i.e. the majority of next hops for most nodes should return legitimate content). Immediate Failover represents the minimum amount of effort a node can expend to avoid bad Data.

In contrast, Probe First is conservative. It requires the node to retain a copy of the Interest that retrieved the bad Data in anticipation of future reports. This Interest will be replayed later as a probe. We believe that producer applications must be able to tolerate replayed Interests as a fact of life in an NDN world because of the multipath nature of the architecture. Additionally, halting forwarding to all possible next hops for a set of namespaces prevents the node and downstream network from being repoisoned until the probe experiment completes.

An attacker can attempt to game Probe First by returning legitimate Data in response to a probe. However, this means that the consumer will (overtime) collect a complete copy of the legitimate content, albeit slowly. Furthermore, this such attack patterns may be detectable and additional mechanisms could be used to experiment with other paths.

It is possible for a node to fail through all of its next hop options in both strategies. Then, the node continues to forward normally (e.g. lowest cost next hop(s)) in the hope that an upstream with more/better options finds legitimate content.

Having every node attempt to avoid poisoned content is an aggressive countermeasure against colluding/compromised network elements. No single node can know for sure whether one of its upstreams is (or is enabling) the bad source. Our goal is for legitimate content to be retrievable for an arbitrary consumer so long as there is no graph cut (malicious node, network failure, etc.) between the consumer and any legitimate content source (endpoint, fragmented cached copies, etc.). We do not consider cached bad Data to constitute a graph cut.

V. Evasion Strategy Evaluation

We evaluate our proposed evasion strategies using ndnSIM [3] on the Sprint PoP Rocketfuel topology [12]. We randomly place varying numbers of consumers and malicious producers with 1 legitimate producer among the topology's 52 nodes. We only consider placement scenarios in which each consumer node has at least one path that does not require traversing a malicious node (i.e. no graph cut). We omit scenarios with more than 1 legitimate producer from our presentation because increasing the count only improves content availability and the time to solution. Altogether, these families of configurations explores some of the worst case scenarios.

Our simulation focuses on *rounds* rather than raw time measurements. Each round begins with the consumer node issuing a normal Data retrieval Interest. We hereafter refer to these Interests as *fetches* to distinguish them from *reports*. If the fetch retrieves malicious Data, then the consumer sends a *report* to notify the network. Each node that receives a report then acts according to the evasion strategy under test. The simulation continues the fetch/report cycle until the fetch returns legitimate Data. Consequently, we omit simulating malicious consumers because they do not impact our metrics. Specifically, the first malicious report would trigger the upstream node to verify, and detect, the attempted deception and result in the consumer being ignored for some policy-based period of time.

We uniformly initialize non-malicious nodes with the Immediate Failover and Probe First evasion strategies. Furthermore, every node uses a slightly modified version of ndnSIM's provided *Best Route* forwarding strategy. ndnSIM's Best Route strategy essentially selects the lowest cost next hop for forwarding. However, it also groups next hops into color coded status: *green* is OK, *yellow* may or may not return content, and *red* is disabled. Green next hops are strictly preferred over yellow ones, regardless of cost. Upon confirming bad Data, our evasion strategies demote a next hop's status to yellow and ensure it has the highest cost (i.e. local max cost + 1). This cost increase has the added benefit of cycling through the next hop options if many failovers are necessary.

ndnSIM's Best Route forwarding strategy promotes a next hop from yellow to green upon the return of any Data packet as described in [14]. However, our experience is that this promotion is harmful when attempting to handle content poisoning attacks; malicious Data packets are indistinguishable from legitimate ones prior to receiving and verifying a report. Consequently, we disable this promotion in Best Route and our Immediate Failover evasion strategy never marks next hops as green.[3] Probe First, on the other hand, can promote next hops because it verifies probe responses.

We performed 10,000 simulation runs per configuration (900,000 total). In 34,560 (4%) these runs, all consumers successfully retrieved legitimate content on their first attempts, thus avoiding the need to send reports or evade bad sources. 26,870 (78%) of these trivial runs occurred in 1 bad producer scenarios compared to 3,424 (10%) for the 20 bad producer configurations. Thus, trivial simulations within a particular scenario configuration represent a relatively small (and decreasing) proportion of runs.

[3] A real implementation could promote next hops after not receiving a report for some period of time.

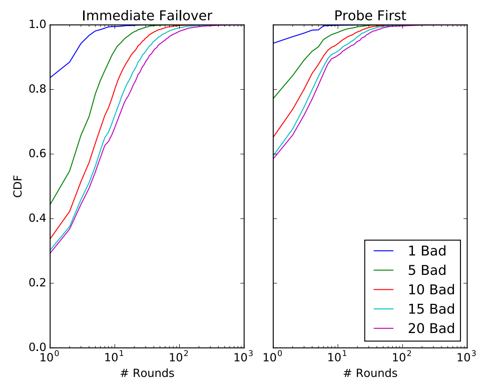

Fig. 1: Number of simulation rounds before retrieving legitimate Data using Immediate Failover.

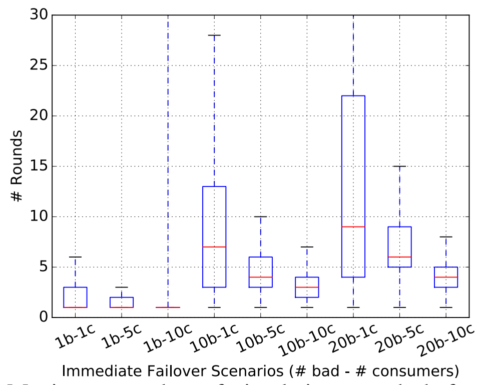

Fig. 2: Maximum number of simulation rounds before retrieving legitimate Data using Immediate Failover.

Figure 1 measures the number of rounds before a consumer node successfully retrieves legitimate Data. We focus on the most extreme scenarios: a single good producer and consumer against up to 20 malicious producers (38.4% of all nodes). Probe First consistently requires fewer rounds than Immediate Failover and has less variance. Furthermore, many Probe First simulations are resolved almost immediately. Intuitively, more malicious nodes require more effort for success. However, even with a large percentage of malicious producers, both strategies often find legitimate content in 10 or fewer rounds. *In the case of Probe First, this occurs 90% of the time*. In other words, evasion strategies can automatically adapt and defeat prefix hijacks despite concerted efforts by an attacker.

Next, we investigate the impact of increasing the number of consumers. Note that we first pruned the rounds in which all consumers trivially found the legitimate producer. We present the number of rounds it takes the last consumer to finish. Figures 2 and 3 show that increasing the consumer count dramatically reduces the range of rounds to solution. Comparing the scenarios with 20 malicious producers (labeled "20b-#c"). Increasing the number of consumers from 1 to 10 reduces the 75th percentile from 22 rounds to 5 (77%) for Immediate Failover and 6 rounds to 2 (66%) for Probe First. Also note that Probe First's 75th percentile is only 40% of Immediate Failover's for one consumer. However, using only 5 consumers yields a 75th percentile of 9 rounds (59%) for Immediate Failover and 3 rounds (50%) for Probe First.

We also analyze the number of poisoned nodes that remain

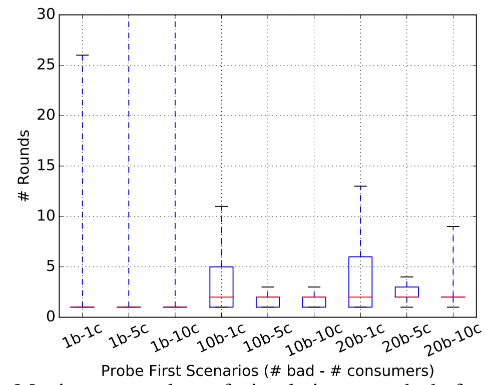

Fig. 3: Maximum number of simulation rounds before retrieving legitimate Data using Probe First.

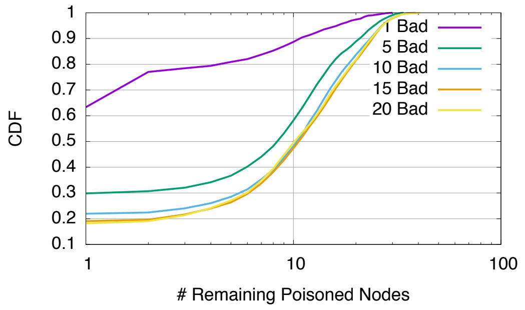

Fig. 4: Number of poisoned nodes remaining after the consumer finds legitimate content using Probe First.

after the consumer has successfully retrieved legitimate Data in scenarios with one good producer and one consumer. We limit our analysis to scenarios with one good producer and one consumer. Immediate Failover, combined with the Best Route strategy, provides a baseline. Each fetch that returns bad Data contributes to its spread in terms of new nodes being exposed. However, the report and cleanup process is symmetric for Best Route; every node exposed to bad Data will be subsequently cleaned up by the next report.

In comparison, Probe First (Figure 4) is asymmetrical because nodes send out additional fetches to probe next hops for paths to legitimate content. These fetches are indistinguishable, and could potentially be duplicates, of the requesting consumer's Interest. Thus, a node's next hops may end up caching and returning bad Data. Our current implementation of Probe First does not report poisoned probe responses because there is no clear halting condition. For example, probing can continuously return bad Data because a nearby node is malicious. Our previous results demonstrate that it may take many rounds to find a legitimate path when there are a significant number of malicious nodes and a next hop could in fact be malicious. Thus, the verification and report process would only serve to increase the probing node's workload unless some other policy was applied. In future work, we will experiment with reporting probe responses a fixed number of times and some back off strategy.

VI. Discussion & Conclusions

Content poisoning is a serious problem for information-centric networks, such as NDN. We propose that the network use namespace ownership rather than attempting to understand application-specific trust models. This puts network elements on equal footing to detect bad Data.

We evaluated two evasion strategies, *Immediate Failover* and *Probe First*, that span the spectrum of effort expended by the network elements to avoid malicious sources. Our simulation results show that both strategies are capable of finding legitimate content; even when nearly half of the topology consists of malicious nodes. More importantly, *these results are achieved automatically*; no operator or other outside intervention is needed to fix what is essentially a prefix hijack. This is a significant change from the current Internet where operators try to use traffic engineering to reclaim hijacked traffic. The key difference is the required, publicly verifiable, signatures on every NDN Data packet. NDN nodes are able to react to problems when signature verification is coupled with the Interest/Data strategy feedback cycle. Thus, we have an idiomatic ICN secure forwarding system that does not need to exchange the concept of "content from anywhere" for traditional secure routing approaches. Our approach highlights the inherent power of secured, named, content.

References

[1] Ndn packet format specification 0.2-alpha-2 documentation.

[2] Alexander Afanasyev. *Addressing Operational Challenges in Named Data Networking Through NDNS Distributed Database*. PhD thesis, University of California Los Angeles, 2013.

[3] Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. ndnsim: Ndn simulator for ns-3. Technical Report NDN-0005, NDN, October 2012.

[4] Alexander Afanasyev, Junxiao Shi, Lan Wang, Beichuan Zhang, and Lixia Zhang. Packet fragmentation in ndn: Why ndn uses hop-by-hop fragmentation. Technical report, NDN, 2015.

[5] Mark Baugher, Bruce Davie, Ashok Narayanan, and Dave Oran. Self-verifying names for read-only named data. *Workshop on Emerging Design Choices in Name Oriented Networking – NOMEN*, 2012.

[6] Paolo Gasti, Gene Tsudik, Ersin Uzun, and Lixia Zhang. Dos and ddos in named data networking. In *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*, 2013.

[7] J. Gersch and D. Massey. Rover: Route origin verification using dns. In *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*, 2013.

[8] Cesar Ghali, Gene Tsudik, and Ersin Uzun. Needle in a haystack: Mitigating content poisoning in named-data networking. In *NDSS Workshop on Security of Emerging Networking Technologies (SENT)*, 2014.

[9] Cesar Ghali, Gene Tsudik, and Ersin Uzun. Network-Layer Trust in Named-Data Networking. *SIGCOMM Computer Communication Review*, 44(5), October 2014.

[10] Dohyung Kim, Sunwook Nam, Jun Bi, and Ikjun Yeom. Efficient content verification in named data networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 109–116. ACM, 2015.

[11] Jun Kurihara, Ersin Uzun, and Christopher A. Wood. An encryption-based access control framework for content-centric networking. *IFIP Networking 2015 Conference*, 2015.

[12] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson. Measuring isp topologies with rocketfuel. *IEEE/ACM Trans. Netw.*, 2004.

[13] Dan Wendlandt, Ioannis Avramopoulos, David G. Andersen, and Jennifer Rexford. Don't secure routing protocols, secure data delivery. In *In Proc. 5th ACM Workshop on Hot Topics in Networks (Hotnets-V)*, 2006.

[14] Cheng Yi, Alexander Afanasyev, Ilya Moiseenko, Lan Wang, and Lixia Zhang Beichuan Zhang and. A case for stateful forwarding plane. *Comput. Commun.*, 2013.

[15] Yingdi Yu. Public key management in named data networking. Technical report, NDN, 2015.

[16] Lixia Zhang, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. Named data networking. *ACM SIGCOMM Computer Communication Review*, 2014.