Computer Science
Technical Report

Mitigating Poisoned Content with Forwarding Strategy

Steve DiBenedetto
Computer Science Dept.
Colorado State University
dibenede@cs.colostate.edu

Christos Papadopoulos
Computer Science Dept.
Colorado State University
christos@cs.colostate.edu

August 13, 2015

Computer Science Department
Colorado State University
Fort Collins, CO 80523-1873

Phone: (970) 491-5792 Fax: (970) 491-2466
WWW: http://www.cs.colostate.edu

Mitigating Poisoned Content with Forwarding Strategy

Steve DiBenedetto
Computer Science Dept.
Colorado State University
dibenede@cs.colostate.edu

Christos Papadopoulos
Computer Science Dept.
Colorado State University
christos@cs.colostate.edu

August 13, 2015

Abstract

Content poisoning attacks are a significant problem in Information Centric Networks (ICN) in general, and Named Data Networking in particular. In content poisoning an attacker inserts bogus content with a legitimate name. While users will reject the content because of signature mismatch, a persistent attacker may use poisoned content as a denial of service attack. While NDN can 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.

In this paper we discuss the problem of content poisoning at some depth and propose two methods to defend against it. We leverage the content verification that users must do anyway and allow them to report bad content. The problem then becomes one of trust between the network and the reporting users so we propose a reputation mechanism that prevents misuse of reports. Then, we propose and evaluate two methods to purge bad content from the network, namely *Immediate Failover* and *Probe First*. These techniques capture the spectrum of possible solutions to purging bad content.

1 Introduction

Content poisoning attacks are a significant problem for information-centric networks (ICNs). In these attacks, malicious routers or end hosts respond to requests with bogus content. Named Data Networking (NDN) is an ICN architecture that requires publicly verifiable signatures on every Data packet so that any node can detect malicious and corrupted content and refuse to accept it.

One way to eliminate poisoned content is for all NDN routers to verify each new piece of content as it passes through. While this maybe possible with future hardware or with dedicated processors, the current pervasive assumption is that in-network signature verification is not practical. The problem is not just computational power: verification also requires key retrieval, which would be prohibitively expensive for routers. The cost becomes even more serious if content poisoning turns out to be a rare event.

Content poisoning is primarily a forwarding problem. Malicious content has been allowed into the network and routers are unknowingly forwarding requests to bad sources. This is different from *cache poisoning*; content poisoning would still exist if there were no in-network caches. Content poisoning has prompted various secure routing (or authorized publishing) schemes as a commonly suggested solution for controlling what content may be injected into the network and from where. However, such schemes are incomplete solutions because bad content may be injected by malicious nodes along the secure path [14].

We believe that traditional secure routing is too restrictive for information-centric networks because secured data, not blessed hosts and the paths to them is the foundation of security. This allows legitimate content to be served from *anywhere*, thus supporting new opportunities [6] and applications.

In NDN poisoned content presents two problems: (1) cached bad content blocks retrieval of good content, and (2) the network may unwittingly prefer a route to a bad publisher. In the first case the network needs to remove bad Data packets from caches and in the second case it needs to forward Interests to a different content source. However, for routers to take action they must detect the bad Data packets, which we dismissed as impractical, at least for now. Conversely, end hosts *must* verify every packet, but unlike routers, cannot make forwarding decisions. The situation presents an opportunity for cooperation between the network and end hosts. We propose that end hosts warn upstream network elements of bad Data and provide the authentication keys required to verify it enabling the upstream elements to independently verify reported problems. Once the problem is confirmed, upstream nodes can remove the bad Data from their Content Store, change forwarding to try alternate paths to good content, and propagate the warning upstream.

In this paper we explore a cooperative approach between the network and the end hosts to detect poisoned content. We present two composable, forwarding strategy modules, termed *evasion strategies* and evaluate their behavior on a real world router topology via simulation with ndnSIM[3]. We find that both evasion strategies automatically find paths to legitimate content and, by extension, defeat a prefix hijack. Furthermore, our strategies do so in 10 or fewer iterations in approximately 70-90% of scenarios, *despite malicious nodes representing nearly half of the network.*

We propose to make in-network verification feasible by shifting the network's understanding of signatures from trust relationships to namespace ownership. We discuss the tradeoffs involved in this change and show that it has minimal, if any, impact on the types of trust models that may be used by applications.

2 Background and Related Work

The Interest's exclusion filter is often proposed (informally and by Ghali *et al* [9]) as a content poisoning mitigation mechanism. The exclude filter allows consumers to enumerate more specific Data packets that cannot satisfy their Interest (e.g. Interest /A Exclude B prevents Data /A/B/* from matching). There are several problems with these types of approaches. First, the knowledge of what to exclude is local to each consumer. Every other consumer must repeat the request and exclude cycle to identify the desired Data. Second, the exclude filter is intended to specify content ranges to eliminate (e.g. versions before 100) rather than enumerating discrete items. An attacker can indefinitely produce malicious Data that collides with legitimate Data names, thus leading to an unscalable exclude filter.

Exclusion is also an important part of the NDN discovery process. For example, applications attempt to identify the exact Data they want (e.g. the latest version). Approaches that rely on exclusion request patterns, such as [9], must ensure that they do not mistakenly flag legitimate exclusion request patterns as indications of bad Data. Latest version discovery in NDN often requires each consumer to exclude the latest Data version to ensure that there is no later version.

Another approach is for Interests to specify the Data's expected signing key. The current NDN packet specification [1] supports a `Key Locator` field in Interests and Data that is either the name of the key or the key's digest. Neither of these options is an effective content poisoning mitigation mechanism because an attacker can simply use the correct key information in malicious Data packets. Ghali *et al* propose the "Interest Keybinding Rule" (IKB) in [10] that requires all Interests to specify the Data's signing key digest and for Data packets to include their entire signing key (rather than 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 in addition to understanding key revocations and application specific trust semantics. However, the end result is an essentially mandatory verification with little meaning; the router has no understanding of whether the Data is legitimate.

Both exclusion and IKB approaches have an implicit assumption, intentional or not, that it is normal for for multiple Data packets to have the same name (excluding their implicit digest). NDN names are intended to be meaningful and express something about the requested/contained content. More strongly, NDN content names are unique before considering the implicit digest [17]. Names are the primary means for differentiating content followed by selector fields such as exclude and key locators.

In this light, name collisions are attacks that attempt to deceive the Interest/Data matching process. We agree with [10] that the network should not become muddled in application specific trust models. Instead, we believe that the network only cares about name *ownership*. If names are unique, there should be some limited set of keys that are authorized to sign for that namespace. These keys may be distributed through some mechanism such as NDNS [2] or CCN-KRS [12]. Then, network elements should be able to traverse a well known hierarchy, for which they have a pre-installed set of anchor keys, to verify any content that is requested of them. Tamper evident revocation loggers are also in development to further aid the verification process [16].

[10] also assumes that routers have and are using alternate paths that deliver legitimate content. However, this depends on the specific forwarding strategy being used 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.

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. IKB's implicit model of multiple keys therefore has serious ramifications for forwarding strategy and potentially the NDN architecture as a whole. Strategy must actively keep track of what (Data,signature) pair is being delivered over each nexthop in order to avoid starving consumers requesting content signed by a different key.

Every NDN Data packet can be represented by a unique hash digest. Consequently, Interests that specify the exact matching Data packet are essentially immune to content poisoning attacks. However, the construction of such Interests faces a bootstrapping problem of determining the exact name. Gasti et al. [7] propose name discovery solutions for static and dynamic content: S-SCIC and D-SCIC, respectively. S-SCIC 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. Inability to pipeline interests leads to a stop and wait retrieval protocol. In comparison, D-SCIC makes use of the `PublisherPublicKeyDigest` (or equivalently, a key locator in the current specification). However, as pointed out by the authors (and above), this field on its own cannot stop malicious Data that merely claims to be signed by a legitimate key.

Gasti et al. [7] 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.

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. 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.

3 Secure Data, Not Routes

Required, publicly verifiable, signatures are the core of NDN's Data packet security. Following the theme of Wendlandt et al. [14], we attempt to use Data signatures as the foundation for a secure forwarding system. Interests and their matching Data, or the lack thereof, form a feedback cycle that can be leveraged by NDN nodes to inform future forwarding decisions by the strategy layer. Assuming an arbitrary NDN node can verify returning Data, the success or failure of verification could therefore be used to adapt the installed forwarding strategy to avoid bad sources.

In-network verification is a contentious topic due to trust issues [10]. The

NDN architecture does not impose a trust model and applications are free to define new models as they see fit. This, in part, leads past work to avoid making in-network trust decisions in favor of approaches such as the *Interest Key Binding* (IKB) rule that leave trust decisions in the hands of the requesting application.

We agree with past work that the network should not attempt to make application-specific trust decisions. However, in-network verification is still a useful 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 whether Alice's trusts Data published by Bob. Instead, the network should only concern itself with whether or not Bob is authorized to publish said Data under the used namespace.* We argue ownership semantics are sufficient for both the needs of applications and the network. 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, similarly, she may not know if Bob is its owner). The network's role is strictly limited to delivering Interests to appropriate destinations. Thus, the network is only concerned with whether Bob is authorized to publish in a particular namespace.

As previously discussed, NDN 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 [8].

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. While hierarchical trust is common, NDN applications also leverage other trust models, such as social network or web of trust, as appropriate to their requirements.

Instead, ownership keys registered with a directory service act as key signing keys and can be used to endorse other, application-specific trust model keys. This effectively glues the application's arbitrary 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 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.

3.1 Improving Key Retrieval

NDN nodes typically verify Data packets through key locator-based retrieval. Each Data packet carries a key locator field that specifies the signing key. However, using this field to guide key retrieval and verification is dangerous because the Data packet cannot be trusted before verifying; an attacker can trivially construct arbitrary length key chains to waste the verifier's resources. Furthermore, the attacker can place said key chains under namespaces it controls and serve the keys itself. Consequently, the attacker can further exploit any key retrieval protocol to maximally inflate fetch time. For example, if the key retrieval protocol allows R retries and each request times out after S seconds, then the attacker can ignore the first $R-1$ Interests and delay the response so long as it arrives in under S seconds. The described attack can occur on each malicious Data packet because there are no namespace binding rules between key locators across packets. Additionally, locator-based verification cannot determine that a Data packet and its key chain are bad until failing to reach a respected trust anchor, which may not exist.

A hierarchical namespace ownership directory greatly simplifies the verification process. In comparison, the retrieval process is *top-down* so the verifying node is always starting from a known anchor. If any subsequently retrieved key fails to authorize, the process can immediately conclude. Furthermore, popular namespaces will likely have theirs, an their parent's, keys cached. Key caching benefits top-down verification more than than bottom-up locator-based retrieval because the verifier can start the process lower in the delegation tree. Bottom-up retrieval must first construct the delegation tree from a, most likely unknown, leaf key until it meets a known key. Again, it is possible that the join point does not exist, thus resulting in wasted key retrieval effort.

4 System Design

NDN requires every Data packet to be signed, but retrieving and verifying keys for 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 and discuss key retrieval in Section 3. 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.[1]

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 mapping to the associated `Upstream-ID`. Reports also identity 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. Thus, we believe it is better for nodes to push the evidence of bad Data.

Network elements 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. At a minimum, this is done by checking the Content Store. However, we envision network elements maintaining a list of recently seen Data exact names (i.e. including SHA-256 digest) and arrival faces to defend against malicious Data packets with no freshness period.[2] 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

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

[2]NDN does not currently have a "do not cache" option.

generates a new report for its own next hop upstream. The bad Data's name and arrival face is then given to the active evasion strategies (see Section 4.1) 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. The exact length of time a downstream should be de-listed 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-whitelisting and a new secret (`Self-ID`) must be shared.

4.1 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. Regardless of the specific problem notification mechanism, 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 can also be used to react to content poisoning attacks once a node becomes aware of the problem.

We developed a set of composable modules called *evasion strategies*. Normally, NDN forwarding strategies choose a subset of next hops for Interest forwarding based on some algorithm (e.g. lowest cost, random, etc.). Evasion strategies are pre-processors for the initial next hop set that attempt to remove or reduce the forwarding preference of next hop choices that lead to bad content sources. The resulting next hop set can then be fed to arbitrary forwarding strategies 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. To the best of our knowledge, we are the first to explore the use of forwarding strategies to mitigate content poisoning attacks.

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. In other words, 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 and essentially the maximum level of effort. Probe First 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 Data 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 and any legitimate content source (endpoint, fragmented cached copies, etc.). We do not consider cached bad Data to constitute a graph cut. Therefore, a consumer should be able to retrieve a legitimate copy when every node is malicious, except for a single path to legitimate content. Furthermore, every node on the legitimate content path may be caching a bad copy and have the worst forwarding preference for the next hop on the legitimate path.

5 Evasion Strategy Evaluation

We evaluate our proposed evasion strategies using ndnSIM [3] on the Sprint PoP Rocketfuel topology [13]. We randomly place 1 consumer, 1 legitimate producer, and vary the number of malicious producers among the topology's 52 nodes. We only consider placement scenarios in which the consumer node has at least one path that does not require traversing a malicious node. Altogether, this family of configurations explores one of the worst case scenarios. Adding additional legitimate producers would provide more opportunities for quickly finding good content. Similarly, reports from additional consumers would be aggregated at path intersections and remove bad Data from the preceding sub-path.

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 [15]. 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 returning probe Interests.

In total, we performed 100,000 simulation runs. 29,408 (29.4%) of these runs successfully retrieved legitimate content on the first attempt, thus avoiding the need to send reports or evade bad sources. 12,314 (approximately 42%) of these trivial runs occurred in 1 bad producer scenarios compared to 3,422 (12%) for the 20 bad producer configurations. Thus, trivial simulations within a particular scenario configuration represent a relatively small (and decreasing) proportion of runs.

Figures 1 and 2 measure the number of rounds before the consumer node successfully retrieves legitimate Data. 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. Even with a large percentage of

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

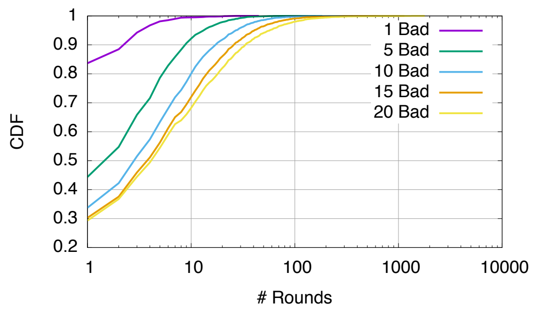

Figure 1: Number of simulation rounds before retrieving legitimate Data using Immediate Failover.

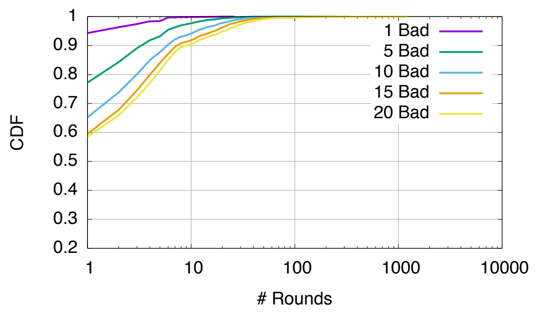

Figure 2: Number of simulation rounds before retrieving legitimate Data using Probe First.

malicious producers (38.4%), 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.

We also analyze the number of poisoned nodes that remain after the consumer has successfully retrieved legitimate Data. As previously stated, Immediate Failover is our baseline strategy because, coupled with Best Route, every returned bad Data packet is cleared by a subsequent report.

Probe First's generated fetches lead to additional nodes becoming poisoned. More bad producers increase the likelihood of a high impact malicious node placement that is highly preferred by others. In most cases, relatively few additional nodes remain poisoned and would be cleared by reports from their respective downstreams.

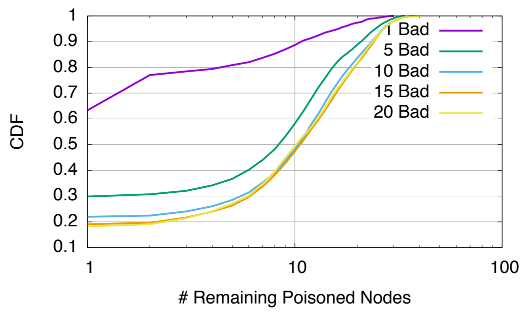

Figure 3: Number of poisoned nodes remaining after the consumer finds legitimate content using Probe First.

6 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] Steve DiBenedetto, Christos Papadopoulos, and Dan Massey. Routing policies in named data networking. *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ICN '11, 2011.

[7] 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.

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

[9] 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.

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

[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] Priya Mahadevan, Ersin Uzun, Spencer Sevilla, and J. J. Garcia-Luna-Aceves. Ccn-krs: A key resolution service for ccn. *ICN*, 2014.

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

[14] 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.

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

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

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