DoS & DDoS in Named Data Networking

Paolo Gasti
School of Engineering and Computing Sciences
New York Institute of Technology
Email: pgasti@nyit.edu

Gene Tsudik
School of Information and Computer Sciences
University of California, Irvine
Email: gts@ics.uci.edu

Ersin Uzun
Palo Alto Research Center
Email: ersin.uzun@parc.com

Lixia Zhang
Computer Science Department
University of California, Los Angeles
Email: lixia@cs.ucla.edu

***Abstract*—With the growing realization that current Internet protocols are reaching the limits of their senescence, several on-going research efforts aim to design potential next-generation Internet architectures. Although they vary in maturity and scope, in order to avoid past pitfalls, these efforts seek to treat security and privacy as fundamental requirements. Resilience to Denial-of-Service (DoS) attacks that plague today's Internet is a major issue for any new architecture and deserves full attention.**

In this paper, we focus on DoS in *Named Data Networking* (NDN) – a specific candidate for next-generation Internet architecture designs. By naming data instead of its locations, NDN transforms data into a first-class entity and makes itself an attractive and viable approach to meet the needs for many current and emerging applications. It also incorporates some basic security features that mitigate classes of attacks that are commonly seen today. However, NDN's resilience to DoS attacks has not been analyzed to-date. This paper represents a first step towards assessment and possible mitigation of DoS in NDN. After identifying and analyzing several new types of attacks, it investigates their variations, effects and counter-measures. This paper also sheds some light on the debate about relative virtues of self-certifying, as opposed to human-readable, names in the context of content-centric networking.

Keywords: Future Internet Architectures; Content-Centric Networks; Named Data Networking; Security; Denial-of-Service.

I. Introduction

Today's Internet is a unique and unprecedented global success story. It serves as a means of disseminating enormous (and ever-increasing) amounts of digital content. Since its inception, the volume of data exchanged over the Internet has witnessed exponential growth. However, core ideas behind the design of today's Internet were developed in the 1970's, when telephony – a point-to-point conversation between two entities – was the only successful example of effective global communication technology. Moreover, original Internet applications were few in number and modest in terms of bandwidth and throughput requirements, e.g., store-and-forward email and remote computer access.

The way people access and utilize the Internet has changed dramatically since the 1970-s and, today, the Internet has to continuously accommodate new services and applications as well as different usage models. To keep pace with changes and move the Internet into the future, a number of research efforts to design new Internet architectures have been initiated in recent years.

Named Data Networking (NDN) [22] is one such effort. NDN is an on-going research project that aims to evolve it into an architectural framework for the future Internet. NDN is also considered an instance of the broader Information-Centric (ICN) approach to networking. NDN explicitly names content (data) instead of physical locations (hosts or network interfaces) and thus transforms content into a first-class entity. NDN also stipulates that each piece of named content must be digitally signed by its producer. This allows for decoupling of trust in content from trust in the entity that might store and/or disseminate that content. These NDN features facilitate caching of content to optimize bandwidth use and enable effective simultaneous utilization of multiple network interfaces.

DoS and DDoS. In recent years, denial of service (DoS) and distributed denial of service (DDoS) attacks have become more common and notorious. In the latter, the adversary exploits a large number of compromised hosts (zombies), that surgically aim their attacks at specific target(s). DDoS attacks are generally easy to instantiate and require little technical sophistication, and they are often very effective and difficult to mitigate [2]. (Hereafter, we use the term *DoS* to indicate both DoS and DDoS attacks.)

We believe that any new Internet architecture should (1) be resilient to existing DoS attacks, or at least limit their effectiveness, (2) anticipate new attacks that take advantage of its idiosyncrasies, and (3) incorporate basic defenses in its design.

To the best of our knowledge, there has been no systematic assessment of how NDN fares with respect to DoS attacks. We believe that such assessment is both timely and important. This work tries to address these issues by analyzing the impact of current DoS attacks on NDN, identifying new attacks that rely on NDN features, and proposing countermeasures. We emphasize that this paper should not be viewed as a comprehensive treatment of DoS in NDN. Instead, it represents a first step towards identifying DoS attacks, assessing their impact and securing NDN against them.

Anticipated Contributions. As mentioned above, our main goal is to identify potential DoS attacks against NDN. By better understanding the nature and effects of these attacks, we can begin to develop feasible and effective countermeasures. Anticipated contributions are as follows:

- We discuss why DoS attacks effective in the current IP-based Internet are largely ineffective against NDN.
- We identify two new types of NDN-specific DoS attacks: *interest flooding*, and *content/cache poisoning*.
- We analyze the impact of several flavors of both attack types and propose a set of potential counter-measures.

Organization: The rest of the paper is structured as follows: Section II overviews NDN. We discuss how NDN copes with existing DoS/DDoS attacks in Section III, and examine new – NDN-specific – attacks in Sections IV and V. Section VI summarizes related work. Finally, we conclude in Section VII.

II. NDN Overview

NDN [22], [16] is a network architecture based on named content. A content name is composed of one or more variable-length components. Component boundaries are explicitly delimited by "/". For example, the name of CNN news content for May 20, 2012 might look like: `/ndn/cnn/news/2012May20`. Large pieces of content can be split into fragments with predictable names. Since NDN's main abstraction is content, there is no explicit notion of "hosts", albeit, their existence is assumed.

Communication in NDN adheres to the *pull* model: A consumer requests content by sending an *interest packet*. If an entity (a router or a host) can "satisfy" a given interest, it returns the corresponding *content packet*. Interest and content are the only two types of packets in NDN. A content packet with name X is never forwarded anywhere unless it is requested by an interest for name X.

NDN routers include the following components:

- *Content Store* (CS), used for content caching and retrieval;
- *Forwarding Interest Base* (FIB), that contains a table of name prefixes and corresponding outgoing interfaces;
- *Pending Interest Table* (PIT), a table containing currently unsatisfied interests and corresponding incoming interfaces;

When a router receives an interest for name X and there are no currently pending interests for X in its PIT, it forwards the interest to the next hop, according to its FIB.[1] For each forwarded interest, a router stores some amount of state information, including the name in the interest and the interface on which it arrived.

Otherwise, if an interest for X arrives while there is already an entry for X in the PIT, the router collapses the incoming interest (and any subsequent interests for X), storing only the interface on which it was received. When content is eventually returned, the router looks up its PIT. If a PIT entry matching the content name is found, the router forwards the content out on all interfaces appearing in such entry. Additionally, it flushes the corresponding PIT entry. If no matching PIT entry is found, then the content is dropped.

Note that, content always follows, in reverse, the path represented by PIT entries in NDN routers. No other information is needed to deliver content. In particular, an interest does not carry a "source address". Any NDN router can provide content caching via its CS. Thus, content might be fetched from any number of in-network router caches, rather than from its original producer. As a result, unlike IP packets, an NDN interest does not include a "destination address".

NDN deals with content authenticity and integrity by making public key signatures mandatory for all content. A signature binds content with its name, and provides integrity, origin authentication and correctness no matter how, when or from where it is retrieved. NDN does not mandate any particular certification infrastructure, relegating trust management to individual applications.

Content objects are named data packets. (We use the terms content object and data packet interchangeably.) Fields of a content object include [5]:

- `Signature`: public key signature (e.g., RSA or ECDSA) computed over the entire content packet, including its name.
- `Keylocator`: references the key needed to verify the content signature. This field can contain one of the following: verification (public) key; certificate containing verification key; or NDN name referencing verification key.
- `PublisherPublicKeyDigest`: hash of the content producer's public key.

In addition to the name of requested content, an interest carries several fields [6]. In this context, only the following are relevant:

- `PublisherPublicKeyDigest`: optional field containing the hash of the producer's public key for the requested content.
- `Exclude`: optional field specifying components that *should not* appear in the name of content returned in response to this interest.
- `AnswerOriginKind`: specifies whether the requested content could be obtained from some routers or must be generated by the producer.
- `Scope`: limits the number of hops an interest can propagate. Scope 0 and 1 limit propagation to the originating host; Scope 2 limits propagation to the first-hop NDN router, etc.

III. NDN and Current DoS Attacks

We now examine some popular types of DoS attacks that work against current TCP/IP-based Internet and assess their putative effects on NDN.

[1]The FIB of an NDN router is similar to that of an IP router, except that it associates name prefixes (rather than IP prefixes) to outgoing interfaces. Also, for a given name prefix a router's FIB may contain multiple interfaces; the NDN routers' *strategy module* makes decisions on how to forward interests according to the FIB and additional external information, such as link congestion.

Bandwidth Depletion. In this common DDoS attack, adversary-controlled zombies flood their victims with IP packets in order to saturate the victims network or server resources. The usual goal is to make the victims unreachable or disable their services. Normally, such attacks are carried out via TCP, UDP or ICMP and rely on sending a stream of packets to the victim at maximum data rate.[2] Similar attacks can be mounted against NDN by directing a large number of zombies to request existing content from a certain victim. However, it is easy to see that the effectiveness of this attack would be limited. Once requested content is initially pulled from its producer, it is cached at intervening NDN routers and subsequent interests retrieve it from these routers' caches. Therefore, the network itself would limit the number of interests that reach the victim.

Reflection Attacks. Instead of letting zombies flood the victims directly, reflection attacks involve three parties: the adversary, the victim host, and a set of secondary victims (reflectors). The goal is to use reflectors to overwhelm the victim with traffic. The adversary creates numerous forged IP packets with the source address set to the address of the intended victim. It then sends these packets to the reflectors. Responses to all such packets are routed to the victim. These attacks require some form of amplification: the amount of content sent by the adversary must be significantly smaller than that received by the victim.

NDN is resilient to reflections attacks due to the symmetric nature of the path taken by each interest and the corresponding content: the latter must follow, in reverse, the path established by the preceding interest. Although an NDN router may broadcast an incoming interest on **some or all** of its interfaces, even under the improbable case where each NDN hop broadcasts the interest, the maximum number of content copies a consumer can receive is bounded by the number of its interfaces. Consequently, the only effective reflection-style attack requires the adversary to be on the same physical network as the intended victim.

Black-Holing by Prefix Hijacking. In a prefix hijacking [3] attack, a misconfigured, compromised or malicious autonomous system (AS) advertises invalid routes in order to motivate other ASes to forward their traffic to it. This can result in so-called "black-holing" whereby all traffic sent to the malicious AS is simply discarded. This attack is effective in IP networks, since, once routing information is polluted, it is difficult for routers to detect, and recover from, the problem. While countermeasures have been proposed (e.g., [19]) this remains a serious threat to the current Internet.

An NDN network has built-in resiliency against prefix hijacking. First, all routing updates are signed and can be verified like all other content packets [14], minimizing prefix hijacking risk except in case of compromised routers. Furthermore, NDN routers have access to strictly more information than their IP counterparts and can use such information to detect anomalies in content retrieval process. Since each content follows the same path as the interest, the number of unsatisfied (expired) interests can be used to determine whether a particular prefix has been black-holed. In addition, NDN routers maintain statistics about performance of each interface with respect to a particular prefix. Loop detection and elimination allows routers to explore topological redundancy through multipath forwarding, enabling NDN routers to try alternative paths as a reaction to perceived attacks.

[2]TCP-based attacks also exploit the connection-based nature of the protocol: each packet sent by zombies tries to open a new connection, which, in turn, requires the victim to create and store corresponding state, thus saturating its resources.

A. Towards New Attacks

Despite ineffectiveness or greatly reduced impact of today's DoS attacks on NDN, variations of aforementioned attacks might be quite effective against NDN.

Recall that an NDN router has two key components that current Internet routers do not have: (1) A Pending Interest Table (PIT) which keeps state to guide content to the requesters, and (2) A Content Store that caches content. Each of them can be subject to malicious attack. One could attempt to deplete PIT entries by Interest Flooding, and to fill the cache with poisoned content. In contrast to current DoS attacks that target end-hosts, Interest Flooding and Content/Cache Poisoning could potentially disrupt content delivery by attacking routers. These new attack types target specific new features of NDN and warrant an in-depth investigation.

IV. Interest Flooding

The adversary can aim at the PIT state in NDN routers to mount an effective DoS attack, which we term *interest flooding*. In this attack, we assume that the adversary uses a large set of zombies to generate a large number of closely-spaced interest packets, aiming to overflow PIT's in routers, preventing them from handling legitimate interests, and/or to swamp the specific content producer(s). Since an NDN interest packet does not carry a source address and is not secured (e.g., not signed) by design, it is not immediately clear how to determine attack origins and take countermeasures.

We can identify three types of interest flooding attacks based on the type of content requested: (1) existing or static, (2) dynamically-generated, and (3) non-existent. While attacks (1) and (3) are mostly aimed at the network infrastructure, (2) can affect both network and content producers.

Similar to bandwidth depletion attacks discussed in the previous section, the impact of type (1) attacks is quite limited since in-network content caching provides a built-in countermeasure. Suppose that there are several zombies, each with independent path to the targeted producer. After the initial "wave" of interests from these zombies, content settles in all intervening routers' caches. Subsequent interests for the same content do not propagate to the producer(s) since they are

satisfied via cached copies[3].

With type (2) attacks, benefits of in-network content caching are lost. Since requested content is dynamic, all interests are routed to content producer(s), thus consuming bandwidth and router PIT state. Also, if generating dynamic content is expensive – signing content is a good example of a relatively expensive per-packet operation – content producers might waste significant computational resources. The direct outcome of this attack type is that the producer may get overloaded with malicious interests, and unable to handle requests from other consumers. The impact on routers varies with their distance from the targeted content producer: the closer a router is to the producer, the greater the effect on its PIT.

Type (3) attacks involve zombies issuing distinct and unsatisfiable interests for non-existent content. Such interests cannot be collapsed by routers, and are routed to the targeted content producers. The latter can quickly ignore such interests without incurring significant overhead. However, these interests will linger and take up space in routers PIT until they eventually expire. We consider routers to be primary intended victims of this attack type. Given an valid name prefix `/prefix`, there are several efficient ways for the adversary to construct multitudes of unsatisfiable interests (that are not easily detectable by routers):

1) Set the name in the interest to: `/prefix/nonce`, where the suffix `nonce` is a random value. Such interests will be forwarded all the way to the producer and never satisfied.
2) Set the `PublisherPublicKeyDigest` field to a random value. Since no public key would match this value, the interest will remain unsatisfied.
3) Set the interest `Exclude` filter to exclude all existing content starting with `/prefix`. Such an interest cannot be satisfied since it simultaneously requests *and* excludes the same content.

A. Tentative Countermeasures

The ease of interest flooding attacks is partly due to lack of authentication of interests. One simple fix might be to require interests to be signed. However, this would immediately raise privacy concerns, as discussed in [9], and would also introduce a new avenue for DoS due to computational overhead of signature verification. On the other hand, we believe that potential problems and DoS attacks due to interest flooding can be addressed without requiring source authentication. Because NDN routers are stateful and can learn much more information about carried traffic than their current IP counterparts. below we discuss two potential countermeasures.

Router Statistics. NDN routers can easily keep track of unsatisfied (expired) interests and use this information to limit the following:

- # of pending interests per outgoing interface: NDN keeps flow balance between interests and content. For each interest sent upstream, at most one content satisfying that interest can flow downstream. Based on that property, each router can compute the maximum number of pending interests per outgoing interface that the downstream connection can satisfy before they time out. Thus, a router should never send more interests than an interface can satisfy, based on average content package size, timeout for interests and bandwidth-delay product for the corresponding link.
- # of interests per incoming interface: Using the same flow balance principle, a router can easily detect when a downstream peer is sending too many interests that cannot be satisfied due to the physical limitations of the downstream link.
- # of pending interests per namespace: When a certain prefix is under attack, intervening routers can easily detect out-of-proportion numbers of unsatisfied interests in their PIT for that prefix. Thus, routers can limit the total number of pending interests for that prefix and throttle incoming interface(s) which has sent too many unsatisfied interests for that same prefix.

It seems challenging to combine the above strategies into one algorithm and choose appropriate parameters for maximum effectiveness against attacks and minimum impairment of legitimate traffic. However our initial investigation has shown strong evidence that one can indeed utilize the per packet state built into each NDN router to enable effective DDoS mitigation [1].

Push-back Mechanisms. A push-back mechanism allows routers to isolate attack source(s). When a router suspects an on-going attack for a particular namespace (e.g., when it reaches its PIT *quota* for that namespace on a given interface), it throttles any new interests for that namespace and reports this action to routers connected on that interface. These routers, in turn, can propagate such information downstream towards offending interfaces, while also limiting the rate of forwarded interests for the namespace under attack. The goal is to push an attack all the way back to its source(s), or at least to a location where it is detectable.

V. Content/Cache Poisoning

We now shift focus to DoS attacks that target content. In this context, the adversary's goal is to cause routers to forward and cache corrupted or fake content, consequently preventing consumers from retrieving legitimate content. We say that a content is *corrupted* if its signature is invalid. Whereas, a content is *fake* if it has a valid signature, however, generated with a wrong (private) key for the name under which it is published.

As mentioned in Section II, each NDN content is signed. Consumers are expected to perform signature verification on all received content. Also, any NDN router can opt to perform signature verification for any content it forwards and caches. Upon receiving and identifying a corrupted or fake content,

[3]Filling caches with arbitrary data requested by the adversary can itself be considered an attack – see, e.g., cache poisoning attack in [29]. In this paper, we do not consider such attacks, since content injected by the adversary is never delivered to consumers who do not request it, nor legitimate content is denied to consumers. To prevents bogus content from filling up caches requires e a good cache management scheme, a topic being actively pursued.

a consumer can request a different copy of the eponymous content using the `Exclude` field in the interest.

In theory, content signatures provide an effective means for detecting content poisoning attacks, since "bad" content can be easily identified and discarded *within* the network, well before reaching consumers. However, in practice, NDN routers face two challenges. The first one is *signature verification overhead*. Our experiments show that routers with multiple Gigabit-speed (or faster) interfaces would need an unrealistic amount of computing power to verify packets at wire rate.

The second challenge is *trust management*, i.e., obtaining the right key to verify a content signature. Although each content includes a reference to its signature verification (public) key, a trust management architecture is needed in order for the routers to determine if a public key is appropriate for a specific piece of content. This creates a tension between flexibility (applications can adopt arbitrary trust models for their content) and security – any NDN router must be able to, if it chooses, verify any content's signature.

Attack Variants. The potential impracticality of NDN routers verifying all signatures on all content opens the door for content poisoning attacks. Although one cannot "push" poisoned content without a prior interest requesting that content, we can identify two attacks variants:

1) The adversary *is aware* of current (pending) interests for particular content, e.g., because it controls some NDN routers. Compromised routers that receive interests for that content simply inject (satisfy interests with) poisoned content, which may then be cached by other intervening routers.
2) The adversary *anticipates* interests in particular content, e.g., a major emerging news-story or about-to-be-released patch for a popular operating system. We also assume that the name of the corresponding content is predictable. The adversary, via numerous distributed zombies, issues many near-simultaneous "legitimate" interests for that content. Next, a compromised host or router (that receives one or more such interests) replies with poisoned content. Then, caches of routers (that processed preceding interests) become populated with poisoned content. Subsequent interests for the same content will return a cached version of the same poisoned content.

While these attack variants require different adversarial capabilities, their impact on the network is almost identical. For this reason, we concentrate on countermeasures that address the *effects* of both.

A. Tentative Countermeasures

First we focus on establishing a strong binding between an interest and a corresponding content. We introduce two constructions, based on standard NDN features, and analyze their benefits and drawbacks. Then, we propose further countermeasures based on heuristics, inter-router communication and consumer feedback.

Self-Certifying Interest and Content

Self-certifying naming [10] (SCN) allows parties to verify the association between a name and a corresponding content without relying on auxiliary information, such as Public Key Certificates and a PKI. This could make SCN an effective countermeasure against content poisoning attacks [12]. There are a few well-known approaches in the literature for implementing SCN. The two most popular ones are geared for static [11] and dynamic content [10], respectively. In the former, an object name is computed as the hash of its content. In the latter, an object name is constructed as: $\mathrm{H}(pk) : L$ where $\mathrm{H}(pk)$ is the hash of the producer's public key pk and L is a human-readable label. Users are not expected to handle self-certifying names directly. Instead, a secure indirection mechanism is needed to map human-meaningful to self-certifying names.

NDN uses hierarchical Human-Readable Naming (HRN). Such names are designed to be user-friendly, i.e., allow consumers to anticipate, guess and remember names of content they wish to retrieve. As discussed in [25], HRN offers a number of advantages over SCN. Unfortunately, human readability precludes a strong (i.e., cryptographic) binding between a name and a corresponding object. In order to determine whether a human-readable name is appropriate for an object, input from a trust management system is required.

We now consider whether it is possible to integrate the functionalities of SCN into NDN without changing the latter's naming structure. To this end, we introduce "Self-Certifying Interests/Content" (SCIC), a mechanism that allows routers to efficiently and securely determine whether a given content is the "correct answer" for a particular interest. We describe two variants of SCIC: one for *static* (S-SCIC) and another for *dynamic* (D-SCIC) content.

Static Content. One component automatically appended to the name of each NDN content is a cryptographic hash computed over its data, name (up to the hash itself) and the signature. A consumer requesting content by name can elect to use this last hash component in an NDN interest. NDN routers can easily and efficiently determine whether a returned content corresponds to its requested name with low overhead. In fact, routers in the current NDN prototype always verify content hashes.

This technique, which we refer to as S-SCIC, prevents the adversary from serving *corrupted* or *fake* content in response to an interest: the hash of the wrong content cannot match the one expressed by the consumer. However it is based on the assumption that the consumer somehow knows the hash of the desired content ahead of time. One way to obtain the hash is to link multiple contents together. For example, let $CO_1, \ldots, CO_m$ be the collection of contents corresponding to a large file. CO_i includes (in its payload) the hash of CO_{i+1}. If the hash of CO_1 can be obtained beforehand, all CO_i-s can be retrieved securely. The problem is thus reduced to discovering the hash of the initial fragment CO_1.

S-SCIC imposes restrictions on inter-content dependencies.

In order for content A to link to content B ($A \rightarrow B$), the latter must be created and named first. This makes it impossible for contents to be linked in a cycle, e.g., $A \rightarrow B \rightarrow C \rightarrow A$. Consequently, it is unclear how to support current Web applications that often involve loops in content linkage. Also, this technique is unsuitable for dynamic content since a consumer has no means of foretelling the hash of a content that does not exist at the time of request, e.g., if the desired content is the result of a Web search.

Clearly, a consumer cannot be expected to anticipate, guess, remember or recognize the hash of the content she is about to request. This translates into a classical *chicken-and-egg* problem. The usual SCN solution is to rely on a *trusted* infrastructure for mapping human-readable to self-certifying names, akin to what DNS does today. Next, we discuss how to address this issue without requiring such infrastructures.

Dynamic Content. As discussed in Section II, an interest can optionally include the `PublisherPublicKeyDigest` field that contains the hash of the public key of the content producer. A consumer can specify the public key that it associates with a desired content name: when this field is present in an interest, each intervening NDN router must make sure that the corresponding content references the same public key. We refer to this technique D-SCIC. Unlike S-SCIC, content that uses D-SCIC can include arbitrary references to other contents, including cyclic links or links to dynamically generated content.

D-SCIC prevents the adversary from injecting *fake* content in response to an interest. However, *corrupted* content may still be returned as long as it references the appropriate producer's public key. This is, again, because NDN routers are not mandated to verify content signatures. However we believe the collaborative verification approach discussed below can effectively mitigate corrupted content.

We believe that a combination of S-SCIC and D-SCIC offers a flexible, trust-model-independent approach for mitigating NDN content poisoning. To the best of our knowledge, no current SCN-based system allows naming content using both "static" and "dynamic" self-certifying names.

Collaborative Verification Techniques

Probabilistic Disjoint Verification. Signature verification load can be evenly distributed among a set of routers belonging to the same organization. Let $r_1, \ldots, r_n$ be NDN routers in the same AS,[4] and let h_{CO} be the least significant 32 bits of the hash of content CO. Router r_i verifies CO if $h_{CO} \equiv i \bmod n$. Assuming that h_{CO} is distributed uniformly between 0 and $2^{32} - 1$, all routers need to verify roughly the same number of contents.

Unfortunately, the adversary can counter-act this strategy by generating contents that are only verified by one router: it picks an arbitrary value $x \in [1, n]$, creates a random content and injects it into the network only if $h \bmod n = x$. To prevent this attack, the hash function used to generate h can be replaced with a keyed hash function (e.g., HMAC), as follows: All routers in the same AS share a secret key k. Let h^k_{CO} be the 32 least significant bits of $\text{HMAC}_k(\text{H}(CO))$. Router r_i verifies CO if $h^k_{CO} \bmod n = i$. Since $\text{HMAC}_k(\cdot)$ behaves like a pseudorandom function, the adversary can succeed with the aforementioned attack only if it learns k.

[4]Routers $r_1, \ldots, r_n$ may belong to different ASs, as long as there is mutual trust between such routers

Neighbor Verification Feedback. By having a large number of routers verifying contents and cooperating, cryptographic operations can be applied less frequently, without lowering the network's resistance to content poisoning attacks.

Each router can verify its cached contents probabilistically and independently. When a router determines that a given content is corrupted, it issues a special *warning* interest on all its interfaces. A warning references `/warning/CO`, where `/warning/` corresponds to a special reserved namespace and `CO` represents the full name (including the hash component) of the corrupted content. The `scope` field of a warning interest is set to 2, i.e., this interest type is not forwarded past one NDN hop.

When a router receives a warning interest, it checks whether its cache contains the referenced content. If not, the router discards the warning. Otherwise it verifies the content with some probability p that might depend on its current workload. If signature verification fails, the router issues its own warning to its neighbors. Otherwise, further warnings from the same interface are ignored for a pre-defined period. To prevent the adversary from injecting fraudulent warnings, every pair of adjacent routers could share a symmetric key and use it to authenticate warnings.

Consumer Feedback. Recall that consumers verify all content signatures. Consumer feedback can be implemented similarly to Neighbor Verification Feedback discussed above, i.e., through specially scoped interests. However, allowing consumers to provide feedback prompts a few new challenges: (1) there may be no pairwise trust relationship between a router and consumers, especially for routers at public access points such as coffee shops or airports; (2) consumers are more likely to be compromised than routers; and (3) it might not be possible to determine which consumer issued a false warning.

One strategy could be based on a probabilistic trust value $T \in [0, 1]$ for each content in a router's cache where the trust value is calculated from explicit consumer feedback. $T = 1$ indicates that the corresponding content has been verified, while $T \approx 0$ indicates that it should be selected for verification with probability proportional to $1 - T$, or deleted if the cache becomes full. A new content is assigned $T = 0.5$. This value increases every time the content is forwarded, and decreases whenever the router receives negative feedback from a consumer. If/when edge routers are not overloaded, an edge router may also verify feedbacks from consumers.

Yet another strategy to aggregate consumer feedback could be doing it implicitly. The expected behavior for consumers receiving corrupted or fake content is to express a new interest that excludes the already received unsatisfactory content using

the exclude field on interests. Routers could monitor what is frequently excluded in consumers interests and rank objects in their CS based on that information. If an interest could be satisfied by more than one object, a router could serve the higher ranked (less excluded) object to satisfy that interest. Cache replacement and selection algorithms for signature verification could also take these rankings into account.

VI. Related Work

Architectures related to Content-Centric Networking include the Data-Oriented Network Architecture (DONA) [18] and TRIAD. DONA is based on "flat" self-certifying names, computed as the cryptographic hash of the producer's public key and a (possibly) human-readable label, which is not cryptographically bound to the content. New content is published (i.e., registered) with a tree of trusted resolution handlers to enable retrieval. Resolution handlers maintain a forwarding table that provides next-hop information for pieces of content in the network. As such, DONA does not support dynamically generated content.

Similar to NDN, TRIAD [8] names content using human-readable, location-independent names. It maps names to available replicas of data using an integrated directory. It then forwards requests until a copy of the data is found. The data location is returned to the client, who retrieves it using standard HTTP. TRIAD relies on trusted directories to authenticate content lookups (but not content itself).

NDN caching performance optimization has been recently investigated with respect to various metrics, including energy impact [24], [20]. To the best of our knowledge, Xie et al. [29] first addressed cache robustness in NDN by introducing CacheShield, a mechanism that helps routers to prevent caching unpopular content, maximizing the use of cache for popular data.

There is a plethora of previous work on DoS attacks on the current Internet infrastructure. Current literature addresses both attacks and countermeasures on the routing infrastructure [15], packet flooding [17], reflection attacks [23] and SYN flooding attacks [28]. Proposed counter-measures are based on various strategies and heuristics, including: anomaly detection [4], ingress/egress filtering [27], IP trace back [21], [26], ISP collaborative defenses [7] and user-collaborative defenses [13].

VII. Summary and Future Work

In this paper, we performed an initial analysis of NDN's resilience to DoS attacks. We started by a brief introduction to the proposed NDN architecture, followed by an examination of the common attacks in today's Internet. The NDN architecture fundamentally changes network delivery from today's "pushing to destinations" to "delivering upon requests", rendering the existing DoS attacks ineffective in an NDN network.

Based on the key components in an NDN router, we identified two new types of attacks that can impact an NDN network: interest flooding and cache/content poisoning, and discussed effects and potential countermeasures.

Clearly, this paper represents only the first step towards mitigation of DoS in the context of NDN. Much more work is needed to evaluate the effectiveness of proposed countermeasures. In particular, extensive simulation- and testbed-based experiments must be conducted in order to determine optimal parameters for the instantiations of our countermeasures.

References

[1] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang. Interest flooding attack and countermeasures in Named Data Networking. In *IFIP Networking 2013*, May 2013.

[2] L. Andersson, E. Davies, and L. Zhang. Report from the IAB workshop on Unwanted Traffic March 9-10, 2006. RFC 4948, Aug. 2007.

[3] H. Ballani, P. Francis, and X. Zhang. A Study of Prefix Hijacking and Interception in the Internet. In *SIGCOMM 2007*. ACM, 2007.

[4] G. Carl, G. Kesidis, R. Brooks, and S. Rai. Denial-of-service attack-detection techniques. *IEEE Internet Computing*, 10(1), jan 2006.

[5] CCNx Content Object. http://www.ccnx.org/releases/latest/doc/technical/ContentObject.html.

[6] CCNx Interests Message. http://www.ccnx.org/releases/latest/doc/technical/InterestMessage.html.

[7] Y. Chen, K. Hwang, and W. Ku. Collaborative detection of ddos attacks over multiple network domains. *IEEE Trans. Parallel Distrib. Syst.*, 18(12), 2007.

[8] D. Cheriton and M. Gritter. Triad: A new next-generation internet architecture.

[9] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun. ANDaNA: Anonymous named data networking application. In *NDSS*, 2012.

[10] D. M. Eres, M. Kaminsky, M. F. Kaashoek, and E. Witchel. Separating key management from file system security. In *In Proc. SOSP*, 1999.

[11] K. Fu, M. F. Kaashoek, and D. Mazieres. Fast and secure distributed read-only file system. In *ACM Transactions on Computer Systems*, 2000.

[12] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker. Naming in content-oriented architectures. In *ICN*, New York, NY, USA, 2011. ACM.

[13] C. Gkantsidis and P. Rodriguez. Cooperative security for network coding file distribution. In *INFOCOM*, 2006.

[14] A. K. M. M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Wang, and L. Zhang. Nlsr: Named-data link state routing protocol. In *SIGCOMM 2013 ICN Workshop*, 2013.

[15] J. Ioannidis and S. Bellovin. Implementing pushback: Router-based defense against ddos attacks. In *NDSS*, 2002.

[16] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking named content. In *CoNext 2009*, 2009.

[17] J. Jun, H. Oh, and S. Kim. Ddos flooding attack detection through a step-by-step investigation. *IEEE International Conference on Networked Embedded Systems for Enterprise Applications*, 2011.

[18] T. Koponen, M. Chawla, B. Chun, A. Ermolinskiy, K. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. *ACM SIGCOMM Computer Communication Review*, 2007.

[19] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang. Phas: a prefix hijack alert system. *USENIX Security*, August 2006.

[20] U. Lee, I. Rimac, and V. Hilt. Greening the internet with content-centric networking. In *e-Energy*, 2010.

[21] L. Lu, M. Chan, and E. Chang. A general model of probabilistic packet marking for ip traceback. In *ASIACCS*, 2008.

[22] Named data networking project (NDN). http://named-data.net.

[23] V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Comput. Commun. Rev.*, July 2001.

[24] E. Rosensweig, J. Kurose, and D. Towsley. Approximate models for general cache networks. In *INFOCOM*, 2010.

[25] D. Smetters and V. Jacobson. Securing network content. PARC, 2009.

[26] R. Stone. Centertrack: an ip overlay network for tracking dos floods. In *USENIX '00*, Berkeley, CA, USA, 2000. USENIX Association.

[27] U. Tupakula and V. Varadharajan. A practical method to counteract denial of service attacks. In *ACSC '03*. Australian Computer Society, Inc., 2003.

[28] H. Wang, D. Zhang, and K. Shin. Detecting syn flooding attacks. In *In Proceedings of the IEEE Infocom*. IEEE, 2002.

[29] M. Xie, I. Widjaja, and H. Wang. Enhancing cache robustness for content-centric networks. In *In Proceedings of the IEEE Infocom*, 2012.