Efficient Content Verification in Named Data Networking

Dohyung Kim
Sungkyunkwan University
Suwon, South Korea
mr.dhkim@gmail.com

Sunwook Nam
Korea Financial Telecommunications and Clearing Institute
Seoul, South Korea
swnam@kftc.or.kr

Jun Bi
Tsinghua University
Beijing, China
junbi@tsinghua.edu.cn

Ikjun Yeom
Sungkyunkwan University
Suwon, South Korea
ijyeom@gmail.com

ABSTRACT

In Named Data Networking, contents are retrieved from network caches as well as the content server by their name. This aspect arises severe security concerns on content integrity. Especially, if poisoned contents lie in the network cache, called content store(CS), interests would be served by the poisoned content rather than they propagate toward the content server. Consequently, users whose interests pass through the contaminated CS cannot access the valid content. In order to resolve the problem, every content is verified before they are inserted into the CS. However, this built-in verification mechanism is not practically feasible due to its huge computational overhead. In this paper, we address problems of content integrity in NDN in details, including how to violate content integrity. We also propose a practical solution that efficiently detects poisoned contents from the CS with minimum overhead. Since the proposed scheme aligns to the basic NDN architecture, it is a practical and effective solution.

Categories and Subject Descriptors

[**C.2.1 Computer Communication Networks**]: Network Architecture and Design

Keywords

Named-Data Networking (NDN); Content integrity; Cache pollution; Verification

1. INTRODUCTION

Recently, Named Data Networking (NDN) [1] has emerged as a promising future network architecture. In NDN, content is addressed by its name, and it is stored anywhere in the network. Hence, users can access the content from either permanent content sources or any temporal network caches, without a prior knowledge of the exact location of the content. This decoupling of content from location enables efficient content distribution by minimizing redundant transmission on the links. However, it brings about new types of security problems as well.

Several NDN specific threats have been introduced in the existing literature [2–7]. Among them, *interest flooding* and *cache poisoning* are discussed as the most representative security attacks. *Interest flooding attacks* explode the number of entries in Pending Interest Table(PIT) of routers, or they dissipate computational resource of the content server by issuing a huge number of closely-spaced interests. In [4, 7], authors present effective schemes to isolate attack sources by analyzing Pending Interest Table. By setting a limit on the number of pending interests that belong to the same namespace, routers efficiently discards malicious interests.

Cache poisoning attacks make routers fill their caches up with fabricated contents. They are classified into two categories in terms of the purpose of attack: 1) *locality attack* that ruins content locality in network caches [5, 6, 8] ; 2) *content poisoning attack* that excludes valid contents from the network. In *locality attack*, the cache file locality is ruined by using unpopular contents. Since every packet is stored in the limited cache space in NDN, popular content can be evicted from the cache by unpopular content. Attackers generate a large amount of unpopular contents, and minimize beneficial effects of in-network caching. In *locality attack*, checking validity of the content itself is meaningless. In [5], authors suggest to cache contents selectively according to their popularity. This scheme prevents insertion of unpopular contents into the cache, which successfully protects popular contents in the cache.

Content poisoning attack isolates valid contents from the network. This type of attack is initiated by placing poisoned content in a router's cache. If an interest whose content name is matched with that of the poisoned content passes through the router, the poisoned content would be served based on simple name matching. While the poisoned content is forwarded back to the user, it contaminates caches of intermediate routers. What is worse, though the user re-requests the content after detecting falsification of the received content, the reissued interest cannot move forward to valid content sources due to the poisoned contents in the network caches.

ICN'15, September 30–October 2, 2015, San Francisco, CA, USA.
© 2015 ACM. ISBN 978-1-4503-3855-4/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2810156.2810165.

Theoretically, *content poisoning attack* is prevented by NDN built-in security mechanism, signature verification. However, signature verification is not mandatory at routers mainly because of its huge computational overhead [9]. The authors in [4] take advantage of hash value of the content to implement probabilistic verification. This scheme successfully reduces verification overhead, but they still have challenges such as inter-packet dependency and trust management. In [9], authors address self-certifying name and consider its limitation. And they introduce a new scheme called 'content ranking' which operates based on users exclusion information. However, user feedback has a potential risk because it can be used to exclude valid contents as well, as mentioned in [10].

In this paper, we discuss *content poisoning attacks* and their impacts in detail, and provide a simple but effective solution for them. Basically, the proposed scheme makes use of built-in signature, but its novelty is to reduce overhead by avoiding unnecessary verification. As previous work [4] points out, a large overhead is expected if every content is checked before being inserted into the cache. In the proposed scheme, verification is performed only for the contents that are actually served from the cache. This strategy gives up the cache integrity temporarily, but it successfully minimizes the effect of poisoned content on the network with the minimum overhead. Via ns-3 simulation, we observe that a large number of contents are evicted from the cache before being reused, even when the cache hit rate is reported very high. Verification for these contents is obviously a waste of resource, which is fundamentally prevented by the selective verification in the proposed scheme. But still, the impact of poisoned content is successfully minimized. In the proposed scheme, the poisoned content is detected before it influences the network, or it is simply evicted from the cache without any malicious effect.

In the implementation of the concept of "check on cache-hit", several delicate situations need to be further considered. First, massive re-requests from attackers trigger too much verification, which degrades router's performance. We solve this problem by introducing the validity mark or applying a segment LRU. If a content is successfully verified, it is stored with the validity mark in the cache. Hence, repeated re-requests for the same content are served without further verification. Besides, a segmented LRU(Least Recently Used) replacement policy is used to favor already-verified contents, which minimizes redundant verification in the cache. Second, poisoned content is distributed over the network by multiple pending interests. In this case, "check on cache-hit" cannot prevent the diffusion of poisoned content. Hence, if a response is matched with multiple pending interests, it is regarded as a cache-hit content and it is verified before being inserted into the cache.

The proposed scheme has similarity with 'check before storing(CBS) [11]' in the sense that popular contents are more likely to be verified. In CBS, content is verified and cached with a certain probability. The probability affects hit ratio as well as recency of network caches. As the probability becomes lower, verification and caching are limited to more popular contents. However, a small probability value makes caches desensitized to the change of content popularity. Hence, it is very challenging to find the optimal probability value in reality. In contrast, the proposed scheme operates upon a cache-hit event, so that it verifies popular contents regardless of the change of content popularity. In addition, the proposed scheme can work together with any type of cache replacement policies, while CBS cannot but stick to the probabilistic caching. Via ns-3 simulation, we show that the proposed scheme minimizes verification overhead up to one thirtieth maximally, when no poison content exists in the network. Since the proposed scheme aligns to the basic architecture of NDN, we believe that the proposed scheme is an effective solution for practical application.

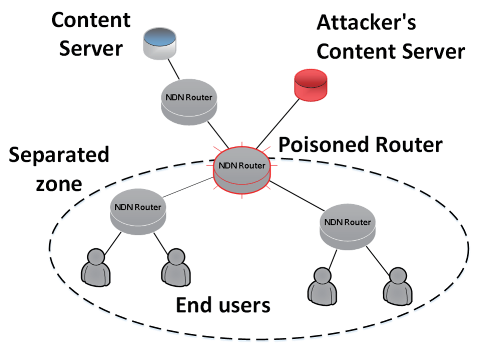


Figure 1: Global poisoning

2. CONTENT POISONING ATTACK

Content poisoning attack is initiated by placing fake content in network caches called content store(CS). It is implemented by using compromised routers or attacker-controlled end hosts. Anyhow, once the poisoned content locates in the CS of a router, future interests for the content may be served by the poisoned content in the CS based on simple name matching. While the poisoned content is delivered to the user, it also contaminates the CS of all intermediate routers. In this way, NDN efficiently distributes the poisoned content all over the network caches. It is one of the distinctive features of *content poisoning attack* that infection is accelerated by the system itself. Another feature of *content poisoning attack*, which is more serious, is to make users separated from the valid content as shown in Fig. 1. If users locate behind the contaminated CS, their requests cannot be forwarded toward the source of valid content. Unless routers check the integrity of cached contents by themselves, NDN architecture is vulnerable to this type of security threats.

Router compromise is an obvious way to contaminate the CS. Besides, *content poisoning attack* is implemented by manipulating end hosts only. Fig. 2 shows one possible scenario. An attacker puts two end hosts within the network. One of them is a client node that issues interests, and the other plays a role of a server that is responsible for injection of poisoned contents into the network. If the client node requests a content, an interest is forwarded to the valid content source by the NDN router, based on Forwarding Information Base(FIB). At this moment, the attacker's server disembogues the poisoned content into the NDN router. Since the NDN router does not check the incoming face of a response, the poisoned content is cached and forwarded back to the user by consuming the pending interest in PIT, as usual. Hence, even when the valid content arrives at the NDN router later, it is simply discarded because no pending interest exists in PIT.

NDN strategy layer brings poisoned content into the network beyond the boundary of autonomous system(AS). In terms of NDN strategy layer, different parts of the content are retrieved from multiple sources in parallel in NDN. This aspect enables fast content retrieval and load balancing. In order to find available sources, authors in [12, 13] suggest an exploration phase where interests are replicated and forwarded via randomly chosen faces in addition to the default face described in FIB. At this phase, interests may be routed to the infected AS region, and poisoned contents come into the out of AS.

Content poisoning attack could be used to implement pharming and phishing attacks, since users are exposed to poisoned content. Besides, network resources (such as link bandwidth and computing resource at routers) are exhausted by repeated transmission of the same interests, which creates a similar effect of DoS (Denial-of-Service) attack.

3. PROPOSED SOLUTION

3.1 Local poisoning

Local poisoning refers to the attack performed within an AS by using end hosts. In local poisoning, poisoned content is injected into the router via different faces from the interest-outgoing face. Hence, local poisoning is prevented by matching the incoming face of a response with the interest-outgoing face. If a response packet does not come from the interest-outgoing face, it should be discarded. Here we note that an initial version of NDN maintains the incoming face of an interest only as a bread-crumb. Fortunately, however, the recent technical report for NFD [14] specifies that the interest-outgoing face is also added to the structure of PIT for implementing various forwarding strategies(Broadcast strategy, Interest retransmission, etc.). Therefore, incoming face of a response can be checked based on PIT entry. It is worth nothing that checking incoming face of a response is necessary only at the edge router that can be directly accessed by end hosts. Therefore, we suggest that the matching functionality is provided by a configurable option.

3.2 Global poisoning

Once the poisoned content has located in the CS, it is globally spread by NDN system itself, which is called global poisoning. In order to prevent global poisoning, poisoned content should be properly eliminated from CS. Otherwise, it serves user requests and contaminates intermediate routers during its delivery. For that, NDN has a built-in security mechanism based on digital signature. However, previous research points out that signature verification incurs too much computational overhead. According to [4], only a throughput of 150Mbps is achieved by an Intel Core 2 Duo 2.53 GHz CPU, although the most convenient RSA public exponent(3) is used.

Surely, signature verification of all relay contents is a big burden to network routers. Here we question *why even the contents that are not actually served should be verified.* Here we use the term ‘serving content’ for the content that is served from the CS, and ‘by-passing content’ for the content that would be evicted from the CS without serving requests. It is obvious that verifying only the serving contents is enough to protect CS from *content poisoning attacks.* Huge computational overhead for by-passing contents is definitely a waste of resource.

We estimate the amount of serving contents in the CS by using ns-3 ndnSIM [15] simulator. In the simulation, 10^5 contents are equally distributed and served by 12 server nodes in Fig. 3. The content popularity follows the zipf-distribution function with parameters, s and $q = 0.7$ [16]. The value of s varies from 0.7 to 1.3. Each client node requests 40 contents per second with exponentially distributed inter-packet delay. The bandwidth of edge links is set to 10Mbps, while that of the core links is 100Mbps. The size of CS is proportional to the link capacity. Each CS stores the amount of contents that arrives for $5 \sim 20$ seconds. The simulation is performed for 1500 seconds.

Fig. 4 shows the hit rate and the proportion of serving contents in the CS at R2 and R7, respectively. ‘r2/r7’ in the legend indicates a router index. The latter part of the legend, ‘0.7/1.0/1.3’, means the value of s in the zipf-distribution. In Fig. 4, it is observed that the proportion of serving contents is much smaller than the value of hit rate. Even though the hit rate is larger than 0.55 in the case of ‘r2, 1.3’, the proportion of serving contents in the CS, is less than 0.1. This is because popular contents in the CS are accessed repeatedly. The result indicates over 90% of traffic is composed of by-passing contents, regardless of hit rate. If we also consider congestion control algorithm that works for filling up the residual link bandwidth, the absolute amount of by-passing contents sustain high regardless of hit pattern.

Based on the observation, the proposed scheme implements the concept of “Check on cache-hit”. In the proposed scheme, every content is stored in the CS without signature verification. Then, if a cache-hit event occurs in the CS, that

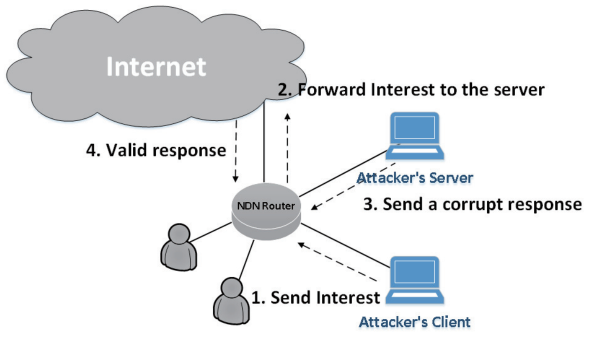


Figure 2: Local Poisoning

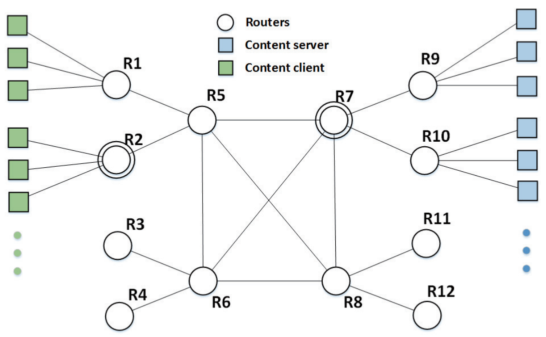
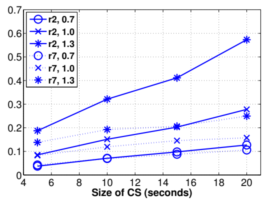
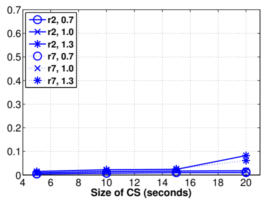


Figure 3: topology



(a) Cache hit rate



(b) Serving content proportion

Figure 4: Proportion of serving contents and cache hit rate

serving content is verified before coming into the network. The proposed scheme can save huge amount of computational resource that has been used for by-passing contents. However, with the reduced number of verification, it successfully minimizes the impact of poisoned content. Under the proposed scheme, poison content in the CS is either simply evicted without any adverse effect, or verified before influencing the network. Here, it is worth nothing that poisoned content can be diffused without the intervention of the CS by multiple pending interests. In order to handle this problem, the proposed scheme regards the content that is matched with multiple pending interests as the serving content, and checks the integrity of the content before storing it, by way of exception.

3.2.1 *Enhancement of the proposed scheme*

In the proposed scheme, if a serving content is successfully verified, CS sets the flag indicating that the content is flawless. The flag is used to avoid redundant verification for the already-verified content at subsequent cache-hits. The flag is valid while the content stays in the CS. It is observed that popular contents are also expelled from the CS due to the limited space, but they are soon re-inserted into the CS. Resultingly, the content is verified repeatedly at each insertion(more precisely, when a cache-hit occurs after each insertion). This inefficiency is also shown in the basic NDN security architecture. To deal with this problem, we apply Segmented Least Recently Used (SLRU) to the CS.

SLRU was originally designed in [17] for efficient disk system. SLRU divides a cache into the protected segment and the unprotected segment, and LRU policy is applied individually for each segment. If an object is hit on the unprotected segment, it is moved to the protected segment and could stay longer than the other objects in the unprotected segment. By giving preference to the frequently referenced objects, SLRU successfully improves a cache hit ratio. If SLRU is applied to our scheme, verified contents moves to the protected segment and cannot be evicted by by-passing contents. Therefore, verified contents have a higher probability of being re-accessed, which successfully lessens the overhead for repeated verification of popular contents.

3.2.2 *Efficiency analysis*

In this section, we analyse the efficiency of the proposed scheme in terms of the metric, κ defined by

$$\kappa = \frac{N_e}{N_v} \tag{1}$$

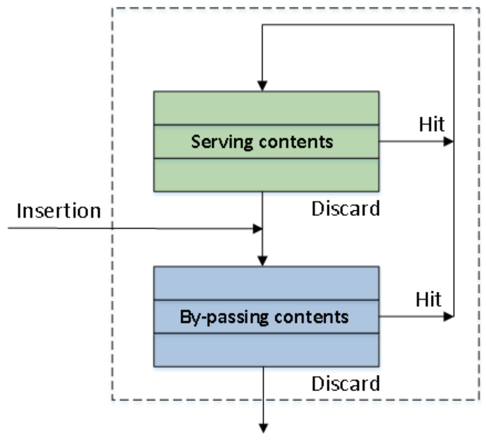


Figure 5: Application of SLRU to the content store

where N_e is the number of examined poisoned contents, and N_v is the number of overall examined contents. The values of κ in the basic scheme, κ_b, corresponds to the ratio of requests for poisoned contents to all requests, e, by definition. The value of κ in CBS, κ_c, is also equal to e because CBS simply controls the amount of N_v in the basic scheme by applying a certain probability, p.

$$\kappa_c = \frac{pN_e}{pN_v} = e \tag{2}$$

In the proposed scheme, verification is performed only for the unverified serving contents. If we let H_p the hit rate for the unverified contents in the CS, N_v for the time interval, Δt, is represented by $Re\Delta t + H_p(1-e)\Delta t$, where R is the request arriving rate. Here we note that a cache-hit occurs when a popular content is re-accessed and a poisoned content is reported by the re-request. N_e is equal to $Re\Delta t$. Hence, the value of κ in the proposed scheme, κ_p, is

$$\kappa_p = \frac{Re}{Re + RH_p(1-e)} = \frac{e}{e + H_p(1-e)} \tag{3}$$

In order to estimate H_p, we assume that the popularity of N total contents follows a "cut-off" Zipf-like distribution. All contents are ranked in the order of their popularity, and $int(i)$ is a request for the i'th most popular content. Then, the conditional probability that the arriving request is $int(i)$, $P_N(i)$, is given by

$$P_N(i) = \frac{\Omega}{i^\alpha} \tag{4}$$

where

$$\Omega = \left(\sum_{i=1}^{N} \frac{1}{i^\alpha}\right)^{-1}$$

α is within the range of $0 < \alpha \leq 1$.

H_p is equal to the probability that the content i that has been inserted into the CS by cache-missed $int(i)$ generates cache-hit for the next $int(i)$. Therefore H_p is represented by

$$H_p = \sum_{\forall i} P_N(i) P_h(i) P_m(i) \tag{5}$$

where $P_h(i)$ and $P_m(i)$ are the probability of cache-hit and cache-miss for the content i, respectively,

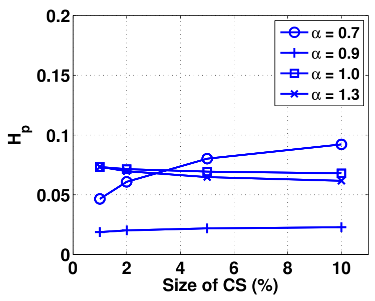
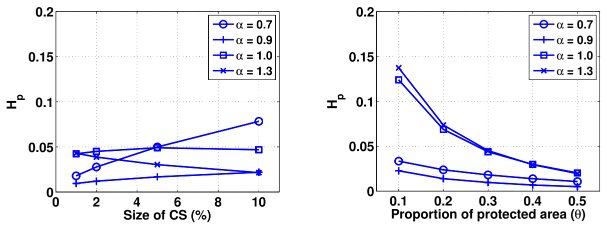


Figure 6: The value of H_p with different cache size



(a) diff. cache size (θ=0.3) (b) diff. θ (1% cache size)

Figure 7: H_p with SLRU

According to Che approximation [18], $P_h(i)$ is represented by

$$P_h(i) = 1 - e^{-P_N(i)t_C} \tag{6}$$

where t_C is the solution of the following equation

$$C = \sum_{\forall i}(1 - e^{-P_N(i)t}) \tag{7}$$

where C is the size of CS.

Based on Taylor series,

$$e^{-P_N(i)t} = 1 - P_N(i)t + \frac{(P_N(i)t)^2}{2} - \frac{(P_N(i)t)^3}{3!} + .. \tag{8}$$

Since $P_N(i)^4 \ll 1$,

$$e^{-P_N(i)t} \approx 1 - P_N(i)t + \frac{(P_N(i)t)^2}{2} - \frac{(P_N(i)t)^3}{6} \tag{9}$$

Hence, t_C is the solution of the following polynomial equation.

$$\frac{(\sum_{\forall i}(P_N^3(i))t^3}{6} - \frac{(\sum_{\forall i} P_N^2(i))t^2}{2} + (\sum_{\forall i} P_N(i))t - C = 0 \tag{10}$$

Fig. 6 shows the value of H_p with different sizes of CS. Even with the CS that can contain 10% of all contents, the value of H_p is less than 0.1.

Therefore, κ_p without SLRU is

$$\kappa_p = \frac{e}{e + H_p(1-e)} \geq \frac{e}{e + 0.1(1-e)} = \frac{e}{0.1 + 0.9e} \tag{11}$$

When SLRU is applied to the proposed scheme, H_p is equal to the conditional probability that content is moved from the unprotected area to the protected area in the CS, given that a new request arrives. That is, H_p corresponds to the hit ratio in the unprotected area. If we let θ the proportion of the protected area in the CS, contents with the rank $i(\leq \theta C)$ are served from the protected area in steady state. Hence, H_p is approximated by (6) and (7) with the normalized popularity distribution for $N - \theta C$ contents and the CS with the size of $1 - \theta C$.

Fig. 7 shows the values of H_p with different θ and C. Since the popular contents are more favored by SLRU, the value of H_p is smaller than that in Fig. 6. As the value of θ increases, smaller values of H_p are resulted because more popular contents can stay in the protected area. Here, it is shown that H_p is smaller than 0.05 with a reasonable size of CS (less than 5%) and $\theta > 0.3$.

$$\kappa_p' = \frac{e}{e + H_p(1-e)} \geq \frac{e}{e + 0.05(1-e)} = \frac{e}{0.05 + 0.95e} \tag{12}$$

Compared with the basic scheme and CBS, the proposed scheme achieves $\frac{1}{0.05+0.95e}$ times larger efficiency. Since $e \ll 1$, $\kappa_p \gg \kappa_b(\kappa_c)$. In CBS, although the value of κ is quite low, the overhead can be controlled by the value of caching probability, p. A smaller value of p lessens the verification overhead, while it helps the CS contain more popular contents. However, if a value of p is too small, it take too long that the CS is refreshed by the recent rising contents. Besides, since CBS is strongly bound together with the probabilistic caching, its application is very limited. By contrast, the proposed scheme performs verification on every cache-hit event, which does not make the CS desensitized to the change of content popularity. And it can work with any well-known caching policies without modification.

4. EVALUATION

In this section, we evaluate the performance of the proposed scheme by using ns-3 ndnSIM simulator. The simulation topology is shown in Fig. 8. 10^6 contents are served by the server node, and their popularity follows the Zipf-Mandelbrot distribution function with parameters value, s. 10 clients request contents with the rate of 40 per second, respectively. Inter-request delays at each client follow the exponential distribution with $\lambda = 0.025$. Link bandwidth is set large enough to exclude the congestion effect. CS is installed only at routers (R1 and R2) and its size is varied from 100 to 2000. Here we note that this paper focuses on how NDN routers detect poisoned contents and delete them from the CS efficiently. Hence, we simply generate poisoned contents at the server with a given error probability. It is beyond the scope of this paper to address how routers isolate malicious nodes and make alternative paths to valid content sources. When clients receive a poisoned content, they immediately re-request the content by issuing a new interest. Simulation is performed for 1000 seconds, and records for the later 500 seconds are analysed.

In the first simulation, we look at the result of verification overhead($\frac{N_v}{N_r}$), and hit rate at R1, when there is no poisoned content in the network. N_r is the number of arriving contents. Caching probability in CBS, p, is set to 0.1. In Fig. 9, the basic scheme shows much larger verification overhead than the other schemes. When $s = 0.7$, most interests are served from the content source rather than CS, which results in low hit rate. Nevertheless, routers verify

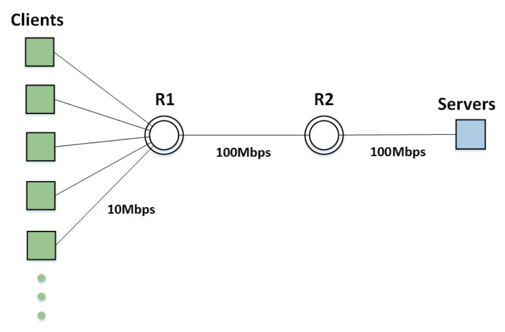
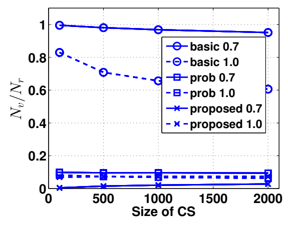


Figure 8: Simulation topology


(a) Verification overhead

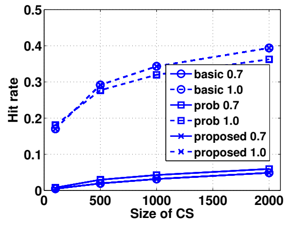
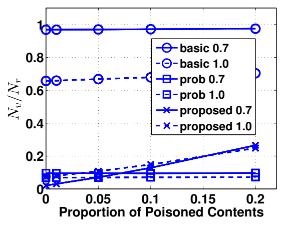

(b) Hit rate

Figure 9: Results without poisoned contents


(a) Verification Overhead

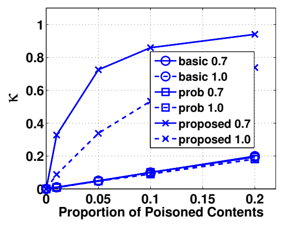
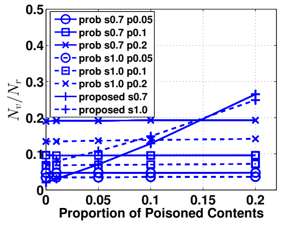
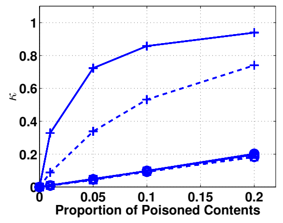

(b) κ

Figure 10: Results with poisoned contents


(a) Verification Overhead


(b) κ

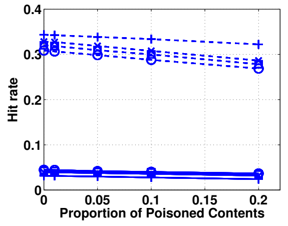

(c) Hit rate

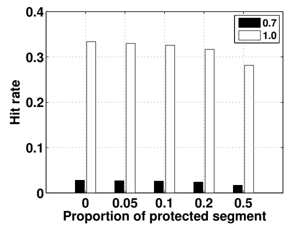
Figure 11: Results with dynamic content popularity

every arriving content in the basic scheme, which makes $\frac{N_v}{N_r}$ stay close to 1. In CBS, caching probability dominantly affects the value of N_v. With $p = 0.1$, N_v comes down to 0.1, and so does the verification overhead. However, caching probability affects the cache hit rate as well. When content popularity becomes less skewed ($s = 0.7$), probabilistic caching with $p = 0.1$ achieves higher hit rate than other schemes. When $s = 1.0$, on the contrary, the smallest hit rate is observed. Since content popularity in the Internet is not fixed and it is changing over time, finding out the proper value of p is not trivial. The proposed scheme achieves the minimum overhead while it does not influence the cache hit rate. Compared with the basic scheme, verification overhead is reduced up to one thirtieth. This result implies that the proposed scheme effectively minimizes verification overhead when there is no poisoned content.

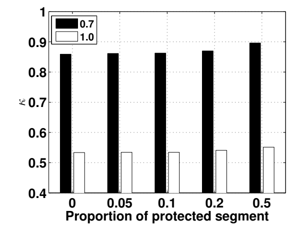
The second simulation is performed with poisoned contents. With the fixed size of CS (1000), we look at the value of verification overhead and κ by varying the amount of poisoned contents in the network. Caching probability in CBS is set to 0.1. Even with different amount of poisoned contents, verification overhead does not change in the basic scheme and CBS, since verification is performed regardless of content's state(See Fig. 10). As in the previous simulation, the largest overhead is shown in the basic scheme, and the verification overhead in CBS is determined by caching probability. In the proposed scheme, however, verification overhead increases in proportion to the amount of poisoned contents. This is because more cache-hits occur in the CS by re-request messages from clients that received poison contents. Here we emphasize that in spite of an increased overhead, the proposed scheme maintains a high value of κ, which indicates unnecessary verifications are effectively minimized in the proposed scheme.

In the third simulation, we compare the proposed scheme with CBS under dynamic content popularity, in terms of verification overhead, hit rate, and κ. We change ranks of content popularity on every 100 seconds. 's0.7/s1.0' in Fig. 11 indicates the value of s in Zipf-Mandelbrot distribution, and 'p0.05/p0.1/p0.2' represents the caching probability, p, in CBS. Like in previous simulations, verification overhead in CBS is managed by the value of p, while the overhead in the proposed scheme increases in proportion to the amount of poisoned contents. However, the proposed scheme obtains a much larger value of κ, which indicates that additional verifications in the proposed scheme result in the detection of poisoned contents. Here we note that caching probability p does not influence the value of κ in CBS, since probabilistic selection for caching is performed regardless of content's state. As concerns the cache hit rate, a smaller value of p results in a higher hit rate when $s = 0.7$. However, as the content popularity becomes more skewed, the benefit of probabilistic caching disappears, and CBS achieves lower hit rates than the proposed scheme.

The fourth simulation is performed to examine the effect of SLRU in the proposed scheme. We set the proportion of poisoned contents as 0.1. The overall size of CS is fixed

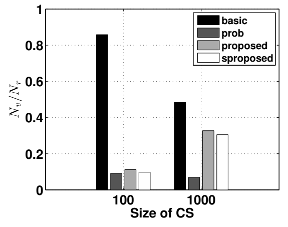


(a) Hit rate

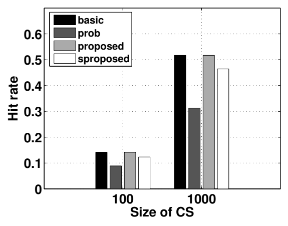


(b) κ

Figure 12: The effect of SLRU



(a) Verification overhead



(b) Hit rate

Figure 13: Results in YouTube trace without poisoned contents

as 1000, and the proportion of the protected segment in CS is varied form 0 to 0.5. Fig. 12 shows the result of hit rate and κ. As the size of the protected segment increases, the time that a content stays at the unprotected segment before eviction becomes shorter. As a result, the cache hit rate decreases. However, due to the protected segments, verification efficiency, κ, grows in proportion to the size of protected segment, which indicates that verified contents are re-used more frequently.

In the last simulation, the proposed scheme is evaluated with the youTube trace from UMASS campus during Mar. 11-17 in 2008 [19]. The overall number of contents in the trace is 158974, and we assume that all contents are of the same size as 1. In Fig. 13(a), it is confirmed that verification overhead is minimized in the proposed scheme (up to less than 25% of the overhead in the basic scheme), when the size of CS is 100. As the size of CS increases, verification overhead increases due to the high hit ratio, but it still remains less than that of the basic scheme. As concerns about the hit rate, the proposed scheme shows the similar performance with the basic scheme in Fig. 13(b). Fig. 14 shows the value of κ when the proportion of poisoned contents is set as 0.1. When the size of CS is 100, the efficiency of the proposed scheme is more than 5 times larger than that of the other comparative schemes. With a larger size of CS, more contents are served from the CS, which increases the number of verification in the proposed scheme. Hence, the value of κ decreases. However, the value of κ is still more than two times larger than that of the comparative schemes when the size of CS is 1000.

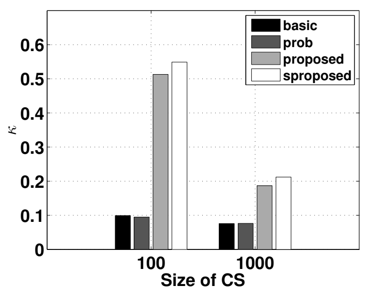


Figure 14: κ in YouTube trace with poisoned contents

5. CONCLUSION

In this paper, we discuss NDN cache poisoning attacks, and provide simple but effective solutions for them. Especially, the proposed solution for global poisoning saves huge amount of computational resource, while it perfectly flushes poisoned contents from the CS. Both simulation and analysis study prove that the proposed scheme effectively mitigates large verification overhead without degradation of CS performance and soundness.

6. ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIP) (NRF-2014K1A1A2064649)

7. REFERENCES

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking named content,” in *ACM CoNEXT*, 2009.

[2] T. Lauinger, N. Laoutaris, P. Rodriguez, T. Strufe, E. Biersack, and E. Kirda, “Privacy risks in named data networking: what is the cost of performance?” *ACM SIGCOMM Computer Communication Review*, vol. 42, pp. 54 – 57, 2012.

[3] M. Wählisch, T. C. Schmidt, and M. Vahlenkamp, “Backscatter from the data plane - threats to stability and security in information-centric networking,” ArXiv abs/1205.4778, Tech. Rep., 2012.

[4] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, “Dos and ddos in named-data networking,” in *IEEE ICCCN*, Aug. 2012.

[5] M. Xie, I. Widjaja, and H. Wang, “Enhancing cache robustness for content-centric networking,” in *IEEE INFOCOM*, 2012.

[6] M. Conti, P. gasti, and M. Teoli, “A lightweight mechanism for detection of cache pollution attacks in named data networking,” *Elsevier Computer Networks*, vol. 57, pp. 3178 – 3191, 2013.

[7] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, “Interest flooding attack and countermeasures in named data networking,” in *IEEE IFIP Networking Conference*, 2013.

[8] Y. Gao, L. Deng, A. Kuzmanovic, and Y. Chen, "Internet cache pollution attacks and countermeasures." in *IEEE ICNP*, 2006.

[9] C. Ghali, G. Tsudik, and E. Uzun, "Needle in a haystack: Mitigating content poisoning in named-data networking," in *NDSS Workshop on Security of Emerging Networking Technologies*, Feb. 2014.

[10] ——, "Network-layer trust in named-data networking," *ACM Computer Communication Review*, vol. 44, pp. 12 – 19, 2014.

[11] G. Bianchi, A. Detti, A. Caponi, and N.Blefari-Melazzi, "Check before storing: What is the performance price of content integrity verification in lru caching?" *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 59 – 67, 2013.

[12] R. Chiocchetti, D. Perino, G. Carofiglio, D. Rossi, and G. Rossini, "Inform: a dynamic interest forwarding mechanism for information centric networking," in *ACM SIGCOMM workshop on Information-centric networking*, 2013.

[13] M. Lee, J. Song, K. Cho, S. Park, T. T. Kwon, J. Kangasharju, and Y. Choi, "Content discovery for information-centric networking," *Elsevier Computer Networks*, 2014.

[14] A. Afanasyev, J. Shi, B. Zhang, L. Zhang, I. Moiseenko, Y. Yu, W. Shang, Y. Huang, J. P. Abraham, S. DiBenedetto, C. Fan, C. Papadopoulos, D. Pesavento, G. Grassi, G. Pau, H. Zhang, T. Song, H. Yuan, H. B. Abraham, P. Crowley, S. O. Amin, V. Lehman, and L.Wang, "Nfd developers guide," NDN Project, Tech. Rep. NDN-0021, Tech. Rep., 2014.

[15] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnsim: Ndn simulator for ns-3," University of California, LosAngeles, Technical Report, Tech. Rep., 2012.

[16] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. shenker, "Web caching and zipf-like distributions: Evidence and implications," in *IEEE INfOCOM*, 1999.

[17] R. Karedla, J. S. Love, and B. G. Wherry, "Caching strategies to improve disk system performance," *Computer*, vol. 27, pp. 38 – 46, 1994.

[18] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: modeling, design and experimental results," *IEEE JSAC*, vol. 20.

[19] "Youtube traces from the campus network," 2008. [Online]. Available: http://traces.cs.umass.edu/index.php/Network