NDN Cache Poisoning Mitigation

Ayush Mattoo, Jyot (Jay) Saini, Yinuo Feng
Instructor : Jianping Pan

CSC 466, Fall 2022

1. Abstract

NDN was proposed as an evolution to current internet architecture, offering a named data alternative to IP packet retrieval strategies. It radically reimagines router architecture and protocols, relying on content stores to speed up packet delivery while still promising security. However, with these new interfaces come new attack surfaces for threat actors to exploit. In this paper, we explore three different cache poisoning mitigation strategies: Brute Force, Smart Selection, and Smart Selection with Limited Hops, ultimately finding that Smart Selection with Limited Hops offers the best theoretical approach to mitigating potential Cache Poisoning attempts.

2. Introduction

The current internet hourglass architecture centers on the universal network layer, which is also known as IP. The design emerged in the 1970s, and the main goal of the architecture was to support host-to-host modules. Gradually, the growth of online shopping, social media, and smartphone applications sharply increased. Thus, the current architecture no longer fits the contemporary needs of the modern internet. Researchers also figured out the current network architecture faces issues with a lack of security, low reliability, lack of mobility, and poor flexibility. In order to solve these questions, five future internet architectures were put forward.

Named Data Networking (NDN) is one of those five future architectures. Its goal is to change the internet by allowing users to request data from a cluster of routers, which handles routing and the distribution of content, in contrast to a single server. This architecture aims to improve data mobility, transfer speeds, and security. However, there are nefarious individuals or groups looking to spread erroneous information using the data-oriented architecture of NDN, and they present new attacks against the framework. These attacks include, but are not limited to, flooding of interest packets attacks, cache pollution attacks, black hole attacks, and, importantly, the attacks that the methods mentioned in this paper will focus on: content poisoning attacks. Consider the following to demonstrate the significance: Cache Hit Ratio–which is a–measurement of how many content requests a cache is able to fill successfully, compared to how many requests it receives.

$$\frac{\text{Number of cache hits}}{(\text{Number of cache hits} + \text{Number of cache misses})} = \text{Cache hit ratio}$$

Fig. 1. The equation for Cache Hit Ratio is below

A Cache Hit Ratio that's higher than 90% can be considered to be able to satisfy most of the requirements of the cache. Based on researcher Abdelhak Hidour and his colleagues, the Cache Hit Ratio should increase as the number of interest packages per second increases. However, as we can see in *Fig. 2*, no matter what kind of cache policies were used, the Cache Hit Ratio will be greatly affected after an attack.

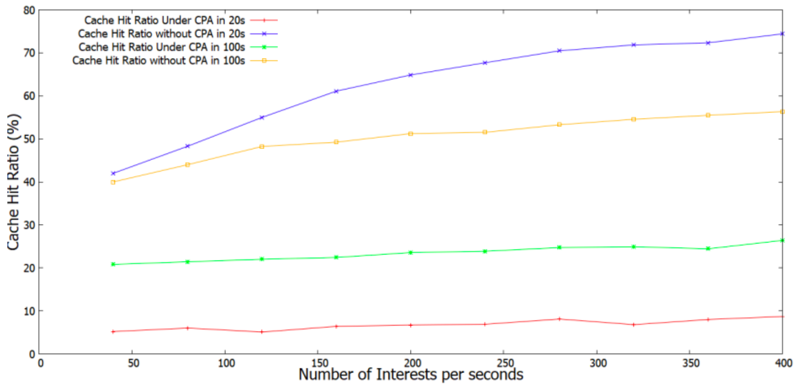


Fig. 2. Number of Interests vs Cache Hit Ratio[3]

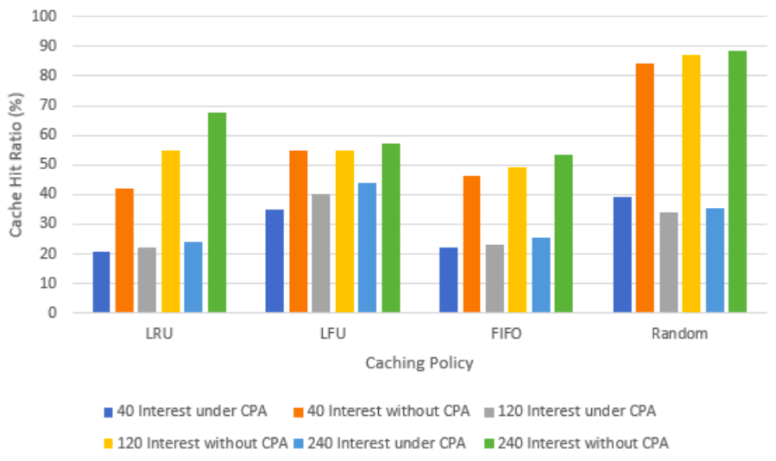


Fig. 3. Interests vs CHR[3]

According to their findings, an NDN Cache Poisoning attacker sending 120 interests per second will cause a 10% drop in delay.

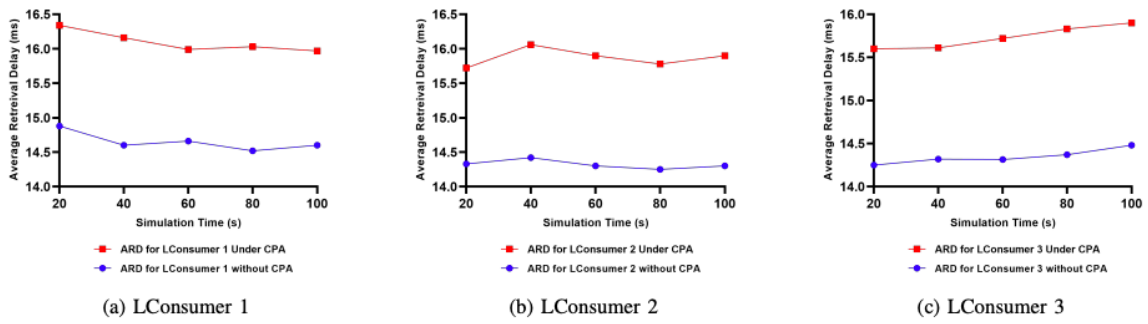


Fig. 4. Average Retrieval Delay[3]

Therefore, the need for NDN cache and content poisoning mitigation strategies is a necessity for security and speed.

3. Related works

There has been a bit of work on cache and content pollution attack [1], [2], and [3]. Likewise, one group of researchers has developed a method for mitigating poisoned data packets using a feedback based technique known as exclusion-based. The paper "Content Poisoning in Named Data Networking: Comprehensive Characterization of Real Deployment" [2] suggests routers can mitigate content poisoning attacks by "verifying data before forwarding" [2], which is a type

1 content poisoning attack whereby routers are compromised. However, they recognize it is impractical for a router to do the calculations, which are very intensive computationally, at efficient enough speeds. Nevertheless, they have determined a goal to accomplish: reducing verification loads on the router by changing the routine the router takes for verification. The strategy they recommend is to accept and cache all the data a router forwards, "but only verifies them when there is a cache-hit"[2].

Type 2 of content poisoning attack is a collaboration between bad providers and clients[2]. The paper outlines a method of content poisoning mitigation leveraging an exclude field where a user can avoid unwanted bad data. This field utilizes the category of content ranking. Where a router ranks its content copies is based on three features that the user specifies to exclude.

(1) Number of exclusions, (2) Time distribution, and (3) Number of exclusions' incoming faces. Overall, copies with a higher rank are more likely to be sent to users. However, since this method relies on clients being trusted and reliable partners, it can be hijacked and manipulated.

Nonetheless, while previous research on content poisoning has been important, it is insufficient, as some solutions involve either eliminating cache entirely or only transferring data from a static cache. These two solutions are not practical or helpful in real-life scenarios. Additionally, we need to consider the aspect of having a false positive rate being high during the detection phase, causing discomfort for all parties involved.

4. Related terms

4.1 - NDN Router

Interests - Interest packets in Named Data Networking are messages sent by a node to request data from another node in the network. The interest packet includes the name of the data requested, which is a unique identifier in the network. It also includes a hop limit and other optional fields such as the minimum and maximum size for the requested data. The interest packet is routed through the network until it reaches the node that holds the data, which then responds with a data packet. This allows data to be requested and retrieved in a decentralized manner, without the need for centralized services or traditional IP addresses.

Data packets - Data packets in Named Data Networking are the basic units of information exchanged between nodes in a NDN network. Each packet contains a name, data, and optional signature and other fields. The name consists of a string of components separated by slashes, which can be used to identify the content of the packet and route it to the correct destination. The data field contains the actual content of the packet, such as an image, video, or text. The signature field is used to authenticate the packet and ensure its integrity. Finally, other optional fields may be included, such as a timestamp or sequence number.

Pending Interest Table - The Pending Interest Table (PIT) is a data structure used in Named Data Networking to store outstanding Interests for which Data has not yet been received. It enables the NDN network to keep track of pending Interests, as well as to forward Interests towards their destinations. The PIT is an important part of NDN's forwarding strategy, as it allows for Interests to be forwarded along multiple paths simultaneously, increasing the chances of receiving Data quickly. The PIT also allows for Interest aggregation, which reduces the number of duplicate Interests being sent in the network.

Content Store - Content Store is a type of data store used to store and manage data that can be accessed by other nodes in the network. It contains a set of data packets, which are indexed by their name, and can be retrieved quickly. Content Store can also be used to store and manage data that is not indexed by its name, such as streaming media. The Content Store is responsible for maintaining the availability of data in the network, and is usually managed by a network node. It is typically implemented as a distributed hash table, which provides a decentralized and robust way of storing data. Content Store is an important part of NDN, as it provides a way for data to be retrieved quickly and securely, without relying on a centralized server.

Forwarding table - A forwarding table in Named Data Networking is a data structure used by an NDN router to determine the next hop for a data packet. It contains a list of prefixes, each associated with a specific forwarding strategy. The forwarding strategy consists of an outgoing interface and a forwarding action, such as dropping, forwarding, or forwarding with a specific hop count. The router uses the forwarding table to decide which interface to use when attempting to forward a packet. The forwarding table is typically populated using a combination of local configuration, information from the NDN naming hierarchy, and routing protocols.

4.2 - Further important terms

Content-Based Routing - Content-Based Routing (CBR) in Named Data Networking is a routing strategy that utilizes the names of the data packets to determine the path they should take through the network. Instead of using IP addresses, NDN uses a combination of the name and the content of the data packet to route it to its destination. Data packets are assigned a name based on the content they contain, and the network maintains a routing table that maps the names of data packets to the routers they should take. This allows the routers to make decisions on how to route the data packets based on the content they contain, rather than using the traditional IP address approach. This can enable more efficient routing and better security, as malicious packets can be identified and blocked more easily.

Faces - Router interfaces are the points of connection between the router and the network. In Named Data Networking, these interfaces provide an interface for forwarding data through the network. A router has many ‘faces’ in an NDN network. Specifically, they provide access to the content store, which stores data packets and forwarding information. They also provide access to the name-based routing table, which is used to determine the best path to a destination. Finally, they provide a means of communication between the router and the NDN nodes, allowing them to exchange control messages

Content/Cache Poisoning - A content positioning is when a router has been attacked, by a compromised router, bad providers, or clients. They push erroneous and tainted data packets to the routers. Subsequently these data packets are forwarded to those that requested them using the interest packet. In addition, in subsequent interests for the tainted data packets, they routers keep forwarding the erroneous data to the consumers. Moreover the data packets can be spread to other routers, causing the erroneous data packets to fan out, rather than be contained in a local area.

5. Content Poisoning Scenarios in NDN

A content poisoning attack in Named Data Networking (NDN) is a malicious attack in which an attacker injects malicious data into the network by falsely claiming it to be legitimate by forging the name of the data. The malicious data is then disseminated throughout the network, and can cause the network's data to be corrupted or even unusable. In order to execute this attack, the attacker may first obtain a copy of the legitimate data, then modify it to include malicious content. The attacker then creates a false version of the data by forging the name of the data. The false version of the data is then injected into the network, and propagated throughout the network. Because NDN uses content-based routing, the malicious data follows the same path as the legitimate data through the network. This means that the malicious data can be propagated to all of the nodes that have access to the legitimate data. Therefore, all of the nodes in the network that receive the legitimate data will also receive the malicious data.

There are many different ways for such a vulnerability to be exploited and in this section we will describe a few different scenarios exploiting the unique properties of NDN:

5.1 Unregistered Remote Provider Scenario

The authors of the paper 'Content Poisoning in Named Data Networking: Comprehensive Characterization of Real Deployment' discovered a vulnerability in NDN's implementation which leads to unspecified behavior. The forwarding component will keep track of the content name and the faces to which the Interest was sent in the PIT entry. However, these out-records appear to be only used for NACK processing. When Data is received, the forwarder only does a PIT match check. This means that Data from any router interface can fulfill the pending Interest, even if it was not sent to that face. Exploiting this flaw, an attacker can deploy an unregistered remote provider by taking control of any client on the route between the clients and the legitimate provider. This will allow the bad provider to send malicious Data that will match pending Interests on the routers towards the clients. The malicious Data will not be aware of the consumer's Interests, but they can still obtain details on recently requested content through a time analysis attack.

When an Interest packet is sent to a core router R2, a race condition begins between the good and bad provider. The first matching Data packet received by R2 will be accepted and will take up the PIT entry, while all the latter ones will be dropped until a new PIT entry is made. This means the malicious Data packet has a higher chance to match an Interest for a targeted content if it arrives at R2 during the time window [$t_{receive}$; $t_{receive} + t_{gpDelay}$]; where $t_{receive}$ is the time when R2 receives the Interest and $t_{gpDelay}$ is the delay of the corresponding Data from the good provider. Since it is difficult for the bad provider to calculate this time window, they must send poisonous Data for targeted contents at a regular rate to increase the success rate of the attack.

5.2 Multicast Forwarding Scenario

When a router using multicast as a forwarding strategy receives an Interest, the Interest is forwarded to all the faces registered in the FIB entry. In the unregistered remote provider scenario, this is done only by the bad provider. However, in the multicast scenario, it requires cooperation from clients in order to pull and insert malicious Data into caches. Clients will send Interests only for the targeted content name, but intentionally not include the current copies of Data in order to go around the caches. This causes the router (R3) to forward the Interests to the next router (R2), which will then forward them to both the legitimate and bad providers due to the multicast forwarding strategy. As a result, Data packets will be sent back by both providers.

The bad Data will arrive first, take up the PIT entry, and be stored in the caches of R2 and R3. The legitimate Data, on the other hand, will be dropped because it was too late to match any PIT entry on R2.

5.3 Best Route Forwarding Scenario

The default forwarding strategy used by the NDN forwarder is to send the incoming Interest to the face with the lowest cost in the FIB entry. This is called the best route forwarding strategy. If more than one face has the same cost, the router will use the first one to be registered. Interests with the same content name, selectors, and a different nonce, will be suppressed if they arrive during a retransmission suppression interval. If an Interest is received after this interval, it is considered a retransmission, and it will be sent to the next lowest cost face that has not yet been used. This allows malicious clients to generate additional Interests and force the router to use a route to the bad provider. This, in turn, enables the bad Data to be cached in R2 and R3. Once all registered faces in the FIB entry have been used, the router sends the Interest back to the first-used face.

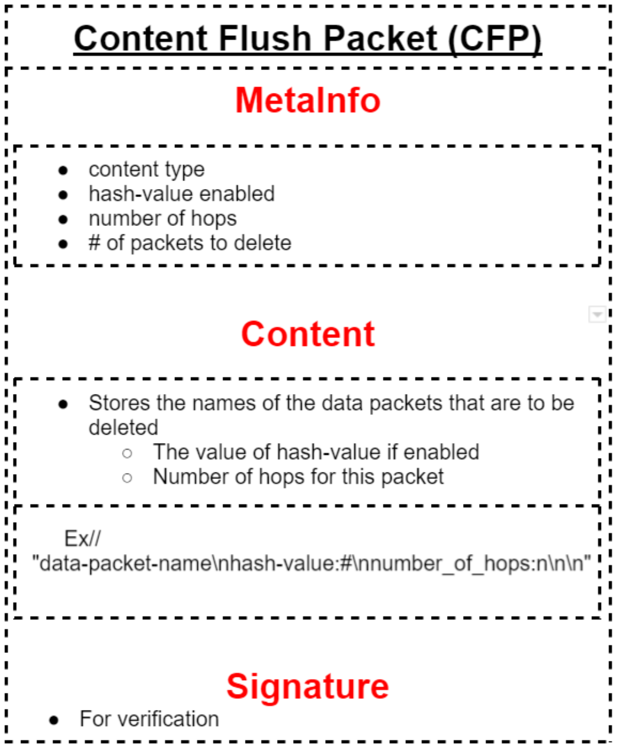
6. Evolution of solution and optimization & Motivation

Named Data Networking which perhaps will be the future of content distribution, will indubitably attract many bad actors. Who hope to hijack by injecting malicious information, intercepting content, manipulating it, and disseminating that data. These exploiters will indeed try to disrupt the end users, the consumers of the content, and this causes a headache for the providers. In the form of security risks and poisoning of data.

The security risks involve diversion of traffic towards fake servers, hijacking of customers' information, and adverse outcomes for both consumers and providers in the form of financial loss or subversion of personal data on the main servers. Poisoning of data, occurs when a malicious actor injects or overrides legitimate data on a NDN node, and subsequently, these data packets are propagated to consumers and other NDN nodes. This tampered data can also potentially simply be the injection of a packet that is unusable, but it can also change files on the computer or forcibly ask for additional interest packets from the NDN router, thereby collecting many packets and creating malware starting from a single data packet.

The motivation for mitigating these back actors is important but also a grand goal; however, it is much needed. In this section, we will focus on a method to limit and possibly eliminate content poisoning in the store. We begin with the brute force method, thereby building a base, and thereafter we are trying to create and implement an efficient yet effective solution for mitigation of content poisoning.

The way to limit content poisoning is to first develop an efficient algorithm to pass off a content_flush_packet(*Fig. 5*), which will traverse the NDN router topology, and consequently each router will find that it has or doesn't have the packet. Subsequently, once it makes that decision, it then passes off the content_flush_packet to other nodes in the NDN topology.



Content Flush Packet (CFP)

MetaInfo

- content type
- hash-value enabled
- number of hops
- # of packets to delete

Content

- Stores the names of the data packets that are to be deleted
  - The value of hash-value if enabled
  - Number of hops for this packet

Ex//
"data-packet-name\nhash-value:#\nnumber_of_hops:n\n\n"

Signature

- For verification

Fig. 5. Content Flush Packet (CFP), which routes to other routers to notify them to purge a certain data packet

6.1 - Network Topology

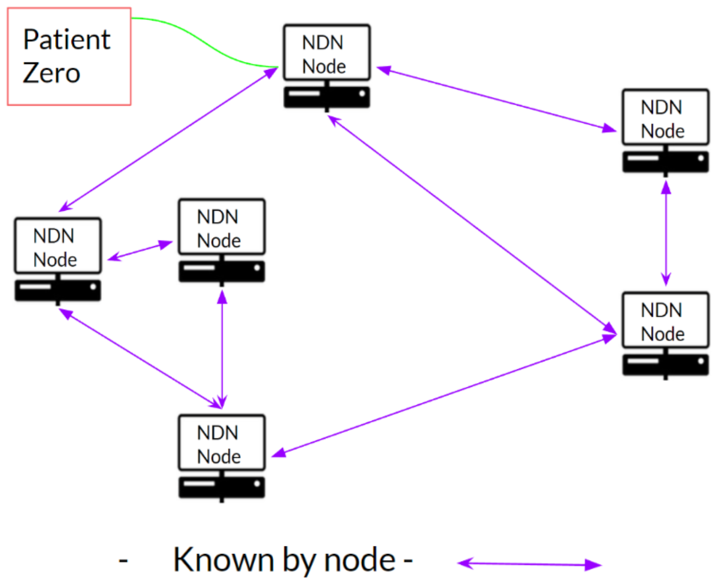


Fig. 6. Network Topology

In the ensuing three algorithms we showcase, all 3 use the same network topology example, where each router will have the same knowledge of its neighboring nodes. In all scenarios, the routers will propagate content_flush_packets, and the node that identifies the content poisoning, will henceforth be known as "Patient-Zero." The Patient-Zero router will begin the propagation of the content_flush_packet. In each case, it will begin by spreading to each node it knows. This design decision was made because, since no node has received the packet yet, it is highly beneficial to spread it to as many nodes as possible in the beginning.

Once a content_flush_packet is received, it will check the content store for the data packet included in the CFP content area. If it is found, it will then delete that data packet. Otherwise, depending on the implementation, it will either continue spreading the CFP or drop the packet. Finally, depending on the implementation, there may be calculations done on the hops remaining.

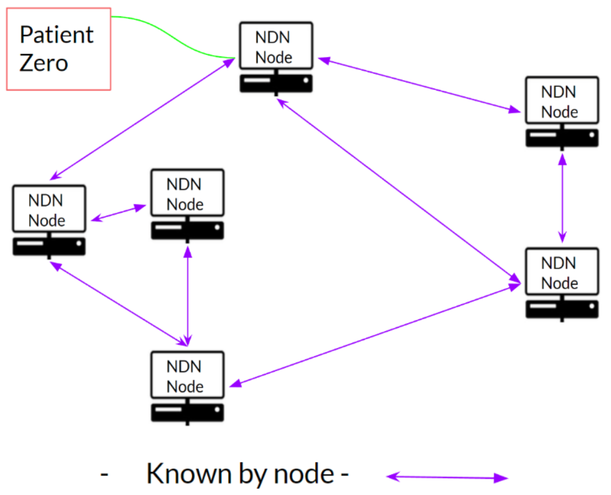


Fig. 7. Network Topology

In the ensuing three algorithms we showcase, all 3 use the same network topology example, where each router will have the same knowledge of its neighboring nodes. In all scenarios, the routers will propagate content_flush_packets, and the node that identifies the content poisoning, will henceforth be known as "Patient-Zero." The Patient-Zero router will begin the propagation

of the content_flush_packet. In each case, it will begin by spreading to each node it knows. This design decision was made because, since no node has received the packet yet, it is highly beneficial to spread it to as many nodes as possible in the beginning.

Once a content_flush_packet is received, it will check the content store for the data packet included in the CFP content area. If it is found, it will then delete that data packet. Otherwise, depending on the implementation, it will either continue spreading the CFP or drop the packet. Finally, depending on the implementation, there may be calculations done on the hops remaining.

6.2 - Brute Force

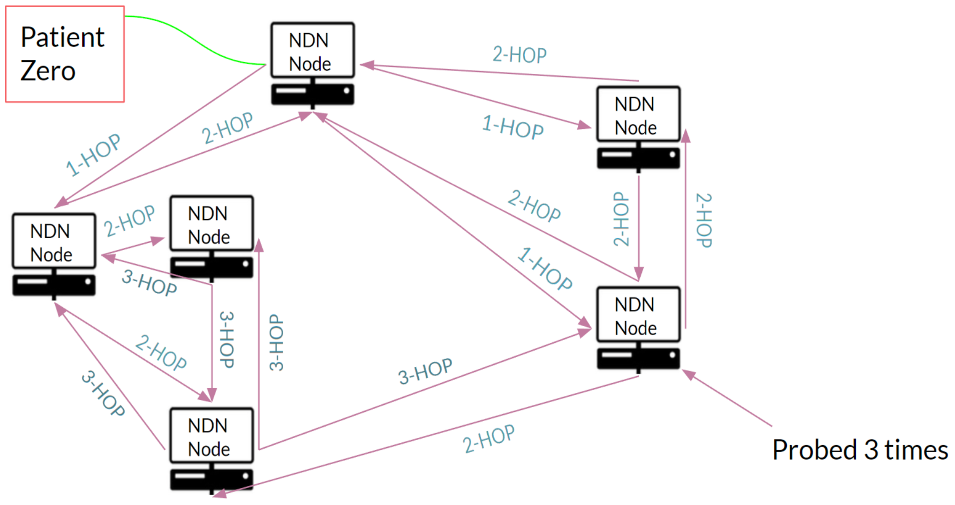


Fig. 8. NDN Network Topology Using Brute Force

We first start off with the most basic method—brute force. For mitigation of content poisoning using the brute force algorithm, the routine is begun by a patient-zero router. It begins by routing the CFP to all the routers it knows; subsequent router transfers will also pass the CFP to every router it knows.

The benefit of this method is important: every node that is affected by the content poisoning—either by alteration or injection of erroneous data packets—will be reached. This has the positive effect of reaching every node that is connected in the topology and saving consumers from bad data. However, there are some obvious unfavorable consequences. Most importantly, the topology is susceptible to flooding. For example, the patient-zero router will send out "n" CFP messages and receive a barrage of "n" messages. It will drop all the incoming packets, but it still has to analyze and examine each and every CFPacket it receives. Subsequently, each router will send out and receive a barrage of CFPs, which will cause a massive strain on the routers themselves.

In a scenario where there are a substantial number of corrupted data packets in the content store, the network topology will be overwhelmed with sending and replying to the CFPs. thereby causing a nuisance for the consumers. Ultimately, the brute force method does solve the problem of a data packet in the CFP being terminated in the topology, but its inefficiency is glaring and best avoided. We consequently want to construct a more efficient algorithm to terminate the erroneous data packets.

6.3 - Smart Selection

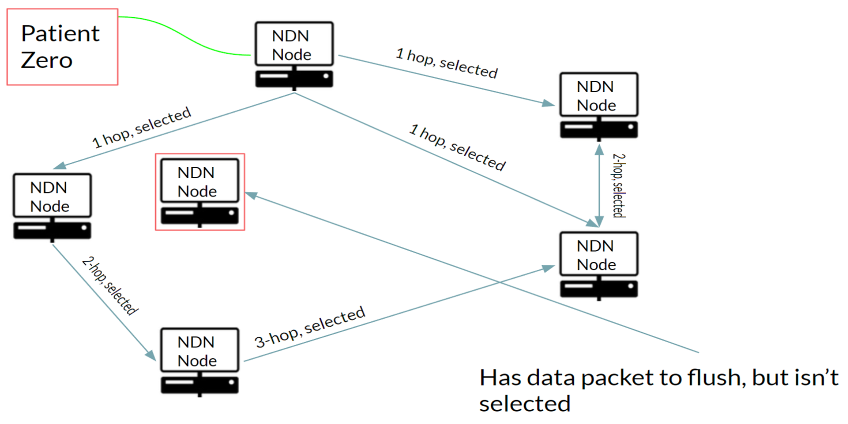


Fig. 9. Network Topology for Limited Selection Using Smart Selection

The second evolution is a method we decided to name the smart selection algorithm. In a nutshell, the goal of the algorithm is to reduce flooding, which we encountered above. The brute force algorithm can be a massive hindrance to the topology in cases of substantial portions of data packets being corrupted in the CS. It accomplishes this by intelligently selecting nodes based on certain criteria, and then using the results of these criteria, it selects a limited number of nodes from the total number of nodes known to the router.

The criteria is based on three components.

I. If a node is known to have access to a larger number of nodes, then pass the CFP to that router.
    A. This ensures a larger subset of the topology can be impacted. Hence, it will best accomplish the goal of mitigating content poisoning.
II. If a node is known to have shared data packets containing the same file path.
    A. This would require keeping an update on the data packets received and from what routers
    B. Such a system has the added benefit of finding if an infected data packet is received from a hijacked node, at which point the router can alert the providers and handlers of NDN.
III. If a node hasn't been communicated in a whole
    A. This is to facilitate communication and make sure smaller nodes are being contacted.

The smart selection algorithm, compared to the brute force method, limits the amount of flooding in the topology, although the flooding will be persistent as it makes its way around. However, this is still a better option, as it helps routers avoid barrages of cache flush packets. Additionally, it never deals with a distinct CFP again once it has encountered one. The negatives of limited selection using smart selection are that it won't reach every affected node, and certain routers might receive a barrage of CFPs.

6.4 - Smart Selection with Limited Hops

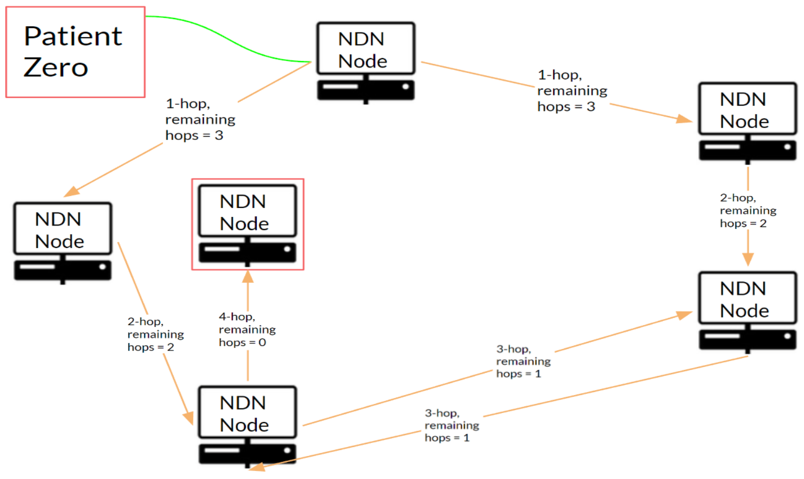

Patient Zero
NDN Node
1-hop, remaining hops = 3
1-hop, remaining hops = 3
NDN Node
2-hop, remaining hops = 2
NDN Node
NDN Node
NDN Node
2-hop, remaining hops = 2
4-hop, remaining hops = 0
3-hop, remaining hops = 1
3-hop, remaining hops = 1
NDN Node
NDN Node

Fig. 10. Network Topology

The final evolution is finding an efficient algorithm to mitigate and reduce the impact of content poisoning. We recommend using the previously mentioned "smart selection" now with the addition of hops. There will be a predetermined, limited number—n. Therefore, the CFP will continue to spread through the routers, until hop counts reach zero.

The benefit of smart selection with a limited number of hops, as we see it, turns out to limit flooding that would have unnecessarily occurred in the topology. Additionally, it greatly reduces, if not completely stops, persistent flooding. As persistent flooding can still occur in a bottlenecked environment.

Most importantly, if a malicious router is spreading erroneous data packets, it is most likely to be in a limited regional or continental range. Thereby, by including a limited number of hops, the topological range that is most affected is targeted, and the routers outside the targeted range are most unimpacted, thereby reducing overhead.

However, there is no free lunch; there are still a few downsides. If hops isn't adjusted for different providers such as Netflix, which is more global, or Hulu, which is concentrated in the United States. The concern is that the CFP might not reach every affected router. Moreover, there needs to be further discussion about the time-to-live for packets so as not to have a packet linger inside a topology.

In conclusion, the evolution of the algorithm from a brute force approach where there is persistent flooding to a limited sporadic burst of flooding. Furthermore, there are two additional mechanisms that are under consideration. The first is to drop the CFP if a router does not have it; the benefit of this is if the poisoning was an isolated incident. The second consideration is to have a packet persist in the topology until the time-to-live has passed, thereby preventing any future erroneous injection.

7. Evaluation and Mathematical Model

Brute force

$$F_{BF} = O((N^2) + \Sigma_{Ln})$$

Smart Selection

$$F_{SS} = O(Y + \Sigma_{Ln})$$

Smart Selection with limited hops

$$F_{SSLH} = O((Y + \Sigma_{Ln}) * Z)$$

Homogeneous Model

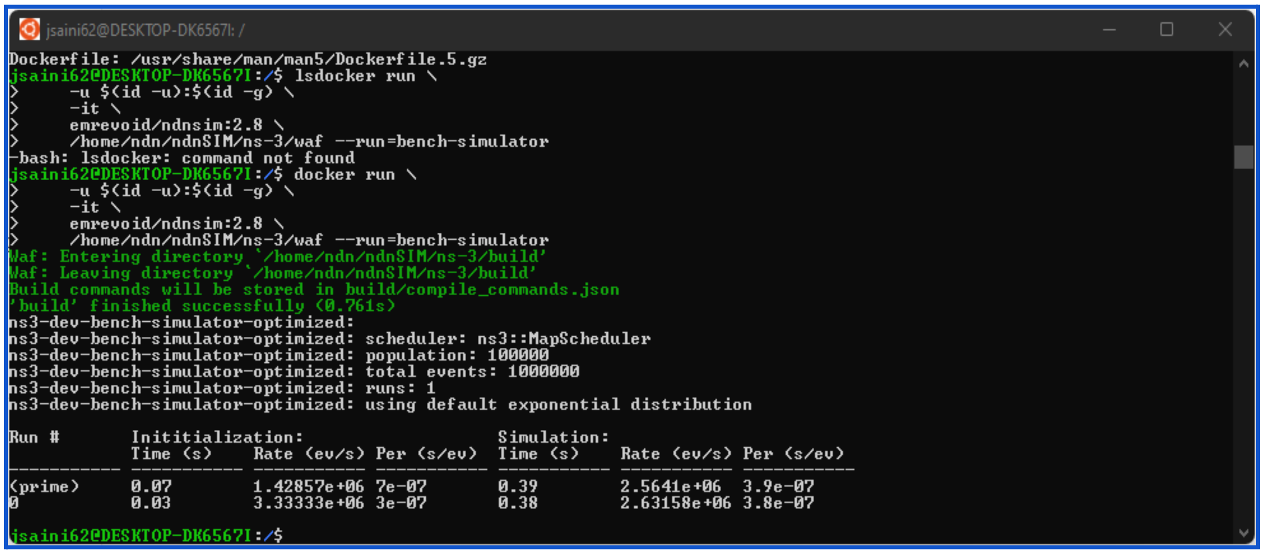
The mathematical model is derived from a homogeneous situation where N, T, X, Y, and Z are constants. The data link's up-and-down speeds are the same. The finding of the mathematical model is to verify that F_{BF} should be avoided. Subsequently, F_{SSLH} is the most efficient way to propagate CFP and limits flooding to a short interval.

Notations	Definitions
F_{BF}	Brute Force
F_{SS}	Smart selection
F_{SSLH}	Smart Selection with Limited Hops
Σ_{Ln}	Total number of hops each originating packet makes from patient zero
N	Total number of active routers in the topology
T	Percent of routers that have that specific cache poisoned packet
X	Delay for one node to another
Y	Total number of routers reached in the propagation of the run of CFP
Z	# number of hops picked (3,4,7, etc..)

8. Results

The results of the simulations, however, were disappointing. We won't go over all of the trials and tribulations we had to go through in order to try and implement our novel solutions in NDNSim, but needless to say, it was not an easy undertaking. More disappointingly, however, it was also a fruitless endeavor as we were unable to have a demo ready in time for the presentation or final report. F_{SS} is more efficient but prone to persistent flooding.

Fig. 11: demonstrates an example of what we would have liked to output from our results by simulating through NDN, except of course we would have liked to track metrics such as the percentage of packets that are poisoned, the percentage of poisoned packets that are rejected, the efficiency of performance gains from cache poisoning mitigation measures against performance losses from our solution's implementation, and more.



```
jsaini62@DESKTOP-DK6567I: /
Dockerfile: /usr/share/man/man5/Dockerfile.5.gz
jsaini62@DESKTOP-DK6567I:/$ lsdocker run \
>     -u $(id -u):$(id -g) \
>     -it \
>     emrevoid/ndnsim:2.8 \
>     /home/ndn/ndnSIM/ns-3/waf --run=bench-simulator
-bash: lsdocker: command not found
jsaini62@DESKTOP-DK6567I:/$ docker run \
>     -u $(id -u):$(id -g) \
>     -it \
>     emrevoid/ndnsim:2.8 \
>     /home/ndn/ndnSIM/ns-3/waf --run=bench-simulator
Waf: Entering directory `/home/ndn/ndnSIM/ns-3/build'
Waf: Leaving directory `/home/ndn/ndnSIM/ns-3/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.761s)
ns3-dev-bench-simulator-optimized:
ns3-dev-bench-simulator-optimized: scheduler: ns3::MapScheduler
ns3-dev-bench-simulator-optimized: population: 100000
ns3-dev-bench-simulator-optimized: total events: 1000000
ns3-dev-bench-simulator-optimized: runs: 1
ns3-dev-bench-simulator-optimized: using default exponential distribution

Run #       Inititialization:                   Simulation:
            Time (s)    Rate (ev/s) Per (s/ev)  Time (s)    Rate (ev/s) Per (s/ev)
----------- ----------- ----------- ----------- ----------- ----------- -----------
(prime)     0.07        1.42857e+06 7e-07       0.39        2.5641e+06  3.9e-07
0           0.03        3.33333e+06 3e-07       0.38        2.63158e+06 3.8e-07

jsaini62@DESKTOP-DK6567I:/$
```

Fig. 11. Example Simulation

9. Further discussion

There is a lot more to explore within the topics we've researched for this paper. For starters, we've barely even touched upon the issue of cache poisoning detection. Mitigating instances of poisoning is well and good, but it is only one part of a two-pronged issue. Detection strategies, while independent of mitigation, can still synergize to make both processes more efficient, or conversely, make both processes more inefficient than they would be if implemented in a vacuum. It's definitely true that any implementation of the proposed solution from previous sections would be dependent on and vary in performance based on the accuracy and precision of the detection strategy implemented.

10. Future work

For further research, our focus would be on heterogeneous testing. where we can manipulate packet loss, data link speed, and the connectivity of each router in the topology. These alterations would allow us to run realistic experiments. Thereby, an accurate mathematical model would be made for the content_flush_packet. This mathematical model would be an equilibrium for various content poisoning scenarios.

- Additionally, we would like to further research the merits of purging the entire data path. If a single data packet's path has been altered or injected by a nefarious router.
- Developing a hash function to be used by routers when comparing CFP. Include this hash value that was produced from the original data packets' content, whereby the received router checks this "hash value" included in the CFP and compares it to the data packet; once it confirms it possesses that data packet. If the hash values are equivalent, then it

drops that data packet; otherwise, it flags that packet and continues its duty to push the CFP to other routers in the topology.

- If a data packet is from a well-known provider like Netflix, Perhaps the limited number of hops will be increased to reach a larger subset of nodes, since the data packet will be affected globally.
- Flagging of a data packet, happens when a router receives a CFP and compares the hash value in the received packet to the freshly calculated value of the data packet. Then it is found that the values are equivalent, but the router still flags the data packet and lets the producer responsible for that packet know of this compromise.
- Finally, perhaps CFP can include additional content flush data names in a single packet, reducing the number of CFPs in the topology and increasing efficiency. The structure would be "data-packet-name\nhash-value:#\nnumber_of_hops:n\n\n" repeated. In addition to this, when a router receives a CFP, it can add on additional data packets that it has found to be contaminated.

11. Conclusion

In this report, we talked about the importance of NDN cache poisoning mitigation, some related works, cache poisoning scenarios, possible mitigation strategies and work done that’s related to those strategies. The strategy we talked about to mitigate the attack is flood nodes that may be affected by the pollution when the attraction was detected. We covered three possible flooding strategies, the first method is the Brute Force method. By using this method, every note that connects to the topology will be flooded. The second method is the limited distribution method. In which only selected nodes and their subset will be flooded. And the last method is called limited distribution with a limited number of hops. This method will further limit the number of times of the hop that will obviously limit flooding and time wasted on this process. Even though we met some problems during the simulation, we still recommend our solution of using cache flood packets and the limited selection with smart selection method.

References

[1] : M. Conti et al., A lightweight mechanism for detection of cache pollution attacks in named data networking, Comput. Netw. (2013)

[2]T. Nguyen, X. Marchal, G. Doyen, T. Cholez and R. Cogranne, "Content Poisoning in Named Data Networking: Comprehensive characterization of real deployment," 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 72-80, doi: 10.23919/INM.2017.7987266.

[3]Abdelhak Hidouri, Mohamed Hadded, Nasreddine Hajlaoui, Haifa Touati, Paul Mühlethaler. Cache Pollution Attacks in the NDN Architecture: Impact and Analysis. SoftCOM 2021 - 29th International Conference on Software, Telecommunications and Computer Networks, Sep 2021, Hvar, Croatia. ffhal03364489f

[4]Afanasyev, A. et al. (2018) A brief introduction to named Data Networking | IEEE Conference ..., A Brief Introduction to Named Data Networking. Available at: https://ieeexplore.ieee.org/document/8599682 (Accessed: December 13, 2022).

[5]Kumar, N. et al. (2019) Security attacks in named Data Networking: A Review and Research Directions - Journal of Computer Science and Technology, SpringerLink. Journal of Computer Science and Technology . Available at: https://link.springer.com/article/10.1007/s11390-019-1978-9 (Accessed: December 13, 2022).

[6]Li, T., Shang, W. and Afanasyev, A. (2018) A Brief Introduction to NDN Dataset Synchronization , IEEE Xplore. Available at: https://ieeexplore.ieee.org/Xplore/guesthome.jsp (Accessed: December 13, 2022).

[7]Conti, M., Gasti, P. and Teoli, M. (2013) A lightweight mechanism for detection of cache pollution attacks in Named Data Networking, Named Data Networking (NDN). Available at: https://named-data.net/publications/ndn_cache_pollution/ (Accessed: December 13, 2022).

[8]Abad, C.L. and Bonilla, R.I. (2007) An analysis on the schemes for detecting and preventing ... - IEEE xplore, IEEE XPLORE. Available at: https://ieeexplore.ieee.org/abstract/document/4279062/ (Accessed: December 13, 2022).

[9]FIU (2019) NDNSIM Archives, Named Data Networking (NDN). Available at: https://named-data.net/tag/ndnsim/ (Accessed: December 13, 2022).

[10]NDN platform documentation (2018) Named Data Networking (NDN). Available at: https://named-data.net/codebase/platform/documentation/ (Accessed: December 13, 2022).

Statement of Contributions

Ayush Mattoo -

Contributed through readings, analysis, research, simulation (settings), video editing and writing contributions for all of our biweekly updates as well as towards the reading summaries on our website.

Jyot Saini -

Contributed through readings, analysis, research, simulation (attempts), and writing contributions for all of our biweekly updates as well as towards the reading summaries on our website.

Yinuo Feng -

Contributed through readings, analysis, research, simulation (settings), and writing contributions for all of our biweekly updates as well as towards the reading summaries on our website.