Cache nFace: uma Contramedida Simples para o Ataque em Conluio Produtor-Consumidor em Redes Centradas em Conteúdo

André Nasserala [1,2], **Igor Monteiro Moraes**[1]

[1]Laboratório MídiaCom, PGC-TCC
Instituto de Computação - Universidade Federal Fluminense
Niterói, Rio de Janeiro, Brasil

[2]Centro de Ciências Exatas e Tecnológicas, CCET
Universidade Federal do Acre
Rio Branco, Acre, Brasil

{anasserala,igor}@ic.uff.br

Abstract. *This paper proposes and evaluates a countermeasure to mitigate the producer-consumer collusion attack in the Content Centric Networking architecture called Cache nFace. The proposed countermeasure mitigates this attack by dividing the cache of a node in sub-caches. Each sub-cache only stores contents requested through a specific network interface. The goal is to increase the robustness of the cache under attack. We perform simulations to evaluate our proposal for different network topologies and attack configurations. Results show Cache nFace reduces the attack efficiency by up to 50% and outperforms other proposal found in the literature in all analyzed scenarios.*

Resumo. *Esse artigo propõe e avalia uma contramedida para comedir o ataque de negação de serviço por conluio produtor-consumidor na arquitetura CCN chamada de Cache nFace. A contramedida proposta comede este ataque dividindo o cache de um nó em sub-caches. Cada sub-cache armazena somente os conteúdos solicitados através de uma interface de rede específica. Isso aumenta a robustez do cache sob ataque. São realizadas simulações para avaliar a proposta para diferentes topologias de rede e configurações do ataque. Os resultados mostram que o Cache nFace reduz em até 50% a eficiência do ataque e supera outra proposta encontrada na literatura em todos os cenários analisados.*

1. Introdução

As Redes Orientadas a Conteúdo são um novo paradigma de comunicação para a Internet [Brito et al. 2012]. O principal objetivo dessas redes é a entrega de conteúdo para os usuários independentemente da localização desse conteúdo, ao contrário da arquitetura TCP/IP, cujo objetivo é a comunicação entre sistemas finais. Diversas arquiteturas foram propostas para esse novo paradigma de comunicação e uma das arquiteturas com maior destaque na literatura é a *Content Centric Networking* (CCN) [Jacobson et al. 2009, Smetters and Jacobson 2009]. Entre as principais características da CCN estão o roteamento através de nomes de conteúdo e o armazenamento de conteúdo em nós intermediários da rede [Jacobson et al. 2009].

Outra característica da CCN é que a segurança é aplicada diretamente aos conteúdos, diferentemente da arquitetura TCP/IP, na qual a segurança é geralmente aplicada ao canal de comunicação entre os sistemas finais [Smetters and Jacobson 2009]. Um pacote de dados CCN é auto-certificado, isto é, ele contém a assinatura digital do conteúdo e do seu nome e a chave pública do produtor, ou informações para obtê-la [Jacobson et al. 2009]. Portanto, qualquer nó de posse de um pacote pode verificar a integridade do pacote e se esse pacote foi gerado pelo produtor que possui tal chave pública. Além disso, a CCN é mais robusta a ataques de negação de serviço (*Denial of Service* - DoS) comuns na Internet atual, como o de esgotamento de banda e o de reflexão, porque na CCN não há garantias de quem um pacote sempre será encaminhado até um dado nó vítima. Isso em virtude do uso de *cache* e da agregação de solicitações de conteúdo feita pelos nós intermediários [Gasti et al. 2013]. No entanto, ataques de negação de serviço específicos para a CCN foram identificados [Smetters and Jacobson 2009].

Um ataque de negação de serviço particular, chamado de conluio produtor-consumidor, entretanto, pode ser efetivo porque os mecanismos nativos empregados pela CCN não são suficientes para inibi-lo. Nesse ataque, consumidores maliciosos solicitam conteúdos que são disponibilizados apenas por produtores maliciosos a uma alta taxa. Isso aumenta o tempo de recuperação de conteúdos legítimos, em virtude do aumento da taxa de erro do *cache* (*cache miss*) para esses conteúdos e, consequentemente, da necessidade de nós legítimos terem que recuperar o conteúdo diretamente do seu produtor. Os mecanismos de segurança da CCN padrão são ineficazes na detecção do ataque em conluio, pois, do ponto de vista da rede, as solicitações e os conteúdos são legítimos. São enviados pacotes de interesse para conteúdos que existem e que são disponibilizados por produtores. O conteúdo é malicioso porque torna popular um conteúdo que não é de interesse de usuários legítimos. Como o produtor malicioso assina os conteúdo de acordo com a política definida pela CCN, os consumidores maliciosos podem solicitá-los sem risco de que esses conteúdos sejam descartados por mecanismos de verificação de assinaturas e chaves.

Em trabalhos anteriores [Nasserala and Moraes 2015, Nasserala and Moraes 2016b, Nasserala and Moraes 2016a], o ataque em conluio produtor-consumidor foi avaliado sem nenhuma proposta de contramedida e em uma topologia de rede específica. Os resultados mostraram que o ataque aumenta o tempo de recuperação de conteúdos em virtude do aumento da ocupação de conteúdos maliciosos e a taxa de erro dos *caches* dos nós da rede. Em um dos cenários avaliados, se 20% dos consumidores são maliciosos e enviam 500 solicitações de conteúdo por segundo cada um, o tempo de recuperação de conteúdos por consumidores legítimos aumenta em 20 vezes. Isso comprova a potencialidade do ataque.

Este artigo, por sua vez, propõe uma contramedida ao ataque em conluio-produtor consumidor chamada Cache nFace. O mecanismo proposto divide o *cache* de um nó em sub-*caches*. Cada um desses sub-*caches* é associado a uma interface de rede de um nó e o espaço de armazenamento de cada sub-*cache* é definido de acordo com taxa de transmissão da interface associada. Dessa forma, um sub-*cache* s_i só pode armazenar conteúdos, cujas solicitações de conteúdo tenham sido recebidas pela interface i. Isso aumenta a robustez do *cache* sob ataque devido à probabilidade de todas as interfaces de um nó serem atacadas ao mesmo tempo ser pequena. Caso exista ao menos uma interface

que não esteja recebendo pacotes maliciosos, essa interface pode responder a solicitações legítimas e aumentar a taxa de acerto do *cache*. Resultados de simulação mostram que o Cache nFace pode reduzir em até 50% o tempo de recuperação de conteúdos legítimos em um dos cenários de ataque avaliados.

O trabalho está organizado da seguinte forma. Na Seção 2 são apresentadas as características principais da CCN e a definição do ataque em conluio produtor-consumidor. Na Seção 3 são apresentados os trabalhos relacionados. Em seguida, na Seção 4, o mecanismo Cache nFace é descrito. Na Seção 5 são definidos os cenários de avaliação e os resultados dos experimentos são analisados e discutidos. Por fim, na Seção 6 as conclusões são apresentadas.

2. A Arquitetura CCN e o Ataque em Conluio Produtor-Consumidor

Esse seção descreve brevemente o funcionamento da arquitetura CCN e define o ataque em conluio produtor-consumidor.

A CCN emprega dois tipos de pacotes: interesse e dados [Jacobson et al. 2009]. Consumidores enviam pacotes de interesse para solicitar um conteúdo. Os produtores ou roteadores de conteúdo respondem aos interesses com pacotes de dados, que carregam o conteúdo em si ou pedaços de conteúdo, chamados de *chunks* [Brito et al. 2013]. Os nós encaminham tanto pacotes de interesse quanto pacotes de dados com base no próprio nome do conteúdo, ao invés do endereço de destino do nó que possui o conteúdo. Um nó recebe um pacote de interesse e responde com o pacote de dados correspondente, ou encaminha o interesse se não tiver o dado em seu *cache*. Para realizar o encaminhamento de pacotes, cada nó CCN tem duas estruturas de dados: a *Pending Interest Table* (PIT) e a *Forwarding Information Base* (FIB). A PIT guarda o estado de cada pacote de interesse encaminhado por um nó que ainda não recebeu uma resposta, ou seja, os interesses que esperam por um pacote de dados. Cada entrada da PIT também armazena as interfaces de recepção de um pacote de interesse e possuem um temporizador associado. É importante ressaltar que o tamanho da PIT é limitado e, dessa forma, novos interesses que chegam enquanto a tabela está cheia não são encaminhados. A FIB atua como uma tabela de encaminhamento para pacotes de interesse. Essa tabela contém uma lista de entradas, cada uma contendo o prefixo de um nome e uma lista de interfaces de saída para as quais os pacotes de interesse com nomes de mesmo prefixo devem ser encaminhados. Os dados são armazenados no *Content Store* (CS), que é efetivamente o *cache* do nó.

Quando um nó CCN recebe um pacote de interesse, ele verifica seu CS para encontrar uma cópia do conteúdo solicitado, cujo nome está no cabeçalho do pacote de interesse. Se o conteúdo está armazenado em *cache*, o nó envia um pacote de dados para o consumidor. Caso contrário, o nó verifica a sua PIT. Se houver uma entrada na PIT para o mesmo conteúdo, o nó atualiza a lista de interfaces de entrada e descarta o pacote de interesse. Caso não haja entrada, consulta a FIB para determinar a interface de saída para encaminhar o pacote de interesse, então, o nó cria uma nova entrada na PIT. Se não houver nenhuma entrada na FIB relacionada com o nome do conteúdo, o pacote de interesse é descartado. Os nós repetem este processo de encaminhamento para cada pacote de interesse recebido. Os pacotes de conteúdo seguem o caminho reverso dos pacotes de interesse, ou seja, são encaminhados por todas as interfaces listadas na entrada da PIT mantida por um nó e que corresponde ao nome do conteúdo [Jacobson et al. 2009].

Ataques de negação de serviço (DoS) são uma ameaça na Internet atual. A arquitetura CCN, entretanto, é mais robusta a esse tipo de ataque do que a pilha TCP/IP em virtude do armazenamento de conteúdo pelos nós intermediários e da agregação de pacotes de interesse [Gasti et al. 2013]. Ataques de esgotamento de banda e de reflexão, por exemplo, são pouco eficientes na CCN. Apesar de ser mais robusta, a arquitetura CCN possui outros ataques e vulnerabilidades identificados [Gasti et al. 2013].

Os ataques de negação de serviço em CCN que esgotam os recursos da rede, afetando a infraestrutura, são classificados, principalmente, em dois tipos: ataques por inundação de interesses ou envenenamento de *cache* [Smetters and Jacobson 2009]. O objetivo dos ataques de inundação de interesses é sobrecarregar a PIT com solicitações de conteúdo enviadas por um nó malicioso a uma alta taxa. Por outro lado, o objetivo do ataque de envenenamento de *cache* é ocupar o *cache* dos nós com conteúdo poluído. Esse conteúdo é enviado por consumidores maliciosos para fazer com que nós armazenem um conteúdo que possua uma assinatura válida, porém corrompido ou aumentem a popularidade de conteúdos menos populares. O ataque em conluio é uma variação do envenenamento do *cache*, cujo objetivo é prejudicar o consumo indireto de conteúdos, isto é, obrigar um consumidor legítimo a recuperar o conteúdo desejado diretamente do produtor. Esse objetivo é alcançado através da manipulação da popularidade dos conteúdos armazenados em *cache*. Consumidores maliciosos enviam pacotes de interesse para um grupo de conteúdos que existem e que são respondidos pelo produtor malicioso. Assim, se solicitado com frequência, um conteúdo se torna popular, apesar de não ter sido solicitado por usuários legítimos. Por isso, o conteúdo é dito malicioso.

O ataque em conluio é possível, porque a CCN emprega políticas de substituição do *cache* baseadas, em sua maioria, na popularidade dos conteúdos. Assim, se um determinado conteúdo não é solicitado com frequência ou não foi solicitado recentemente pelos consumidores, ele é considerado menos popular. Dessa forma, terá prioridade de descarte quando houver necessidade de armazenar novos conteúdos. Ao solicitar um conjunto específico de conteúdos e em taxas altas, os nós maliciosos manipulam a política de substituição do *cache*. Com mais conteúdos maliciosos em *cache*, maior a taxa de erro para os conteúdos legítimos e, consequentemente, maior a necessidade de nós legítimos terem que recuperar o conteúdo diretamente do seu produtor. Mesmo que os consumidores legítimos não tenham que consumir diretamente dos produtores, eles terão seus interesses encaminhados por mais saltos até conseguir o conteúdo desejado.

Uma das principais razões para que o ataque em conluio consumidor-produtor seja bem sucedido é o fato de que os pacotes de interesse e de dados usados no ataque são legítimos para a rede e, portanto, não são detectados por mecanismos de verificação de assinaturas. O pacote com o conteúdo malicioso possui uma assinatura válida, carrega a chave do publicador, ou informações para obtê-la, e assim, passa no teste de verificação de integridade e autenticidade. Logo, não é identificado como malicioso e nem descartado.

3. Trabalhos Relacionados

Todos os trabalhos descritos a seguir avaliam e/ou propõem contramedidas para os ataques de negação de serviço por inundação de interesses e por envenenamento de *cache*. Porém, nenhum, especificamente, propõe uma contramedida para comedir ataques em que consumidores e produtores maliciosos agem em conluio para gerar, disponibilizar

e manipular a popularidade de conteúdos. O mecanismo proposto nesse trabalho visa comedir esse tipo de ataque, como detalhado na Seção 4.

Gasti *et al.* propõem um mecanismo de *push-back* como contramedida ao ataque de inundação de interesses [Gasti et al. 2013]. Esse mecanismo monitora a ocupação da PIT e identifica quando uma determinada interface está próxima de atingir seu número máximo de entradas. Desta forma, o mecanismo avalia e controla o fluxo de pacotes de interesse que contém os mesmos nome de prefixos para determinar um limiar. Além disso, a contramedida envia uma notificação na interface supostamente atacada que será recebida por um nó vizinho. Esse nó, por sua vez, deve propagar tal informação no sentido das interfaces atacadas e, ao mesmo tempo, limitar a taxa de interesses encaminhados que contenham o prefixo sob ataque. Portanto, o objetivo da contramedida é empurrar o ataque para o caminho de volta até o atacante, ou pelo menos para um nó no qual seja detectado [Gasti et al. 2013]. A principal característica dessa contramedida é não modificar a arquitetura padrão proposta para a CCN. O ponto fraco do trabalho de Gasti *et al.* é que nem o impacto do ataque e nem a contramedida proposta são avaliados por simulação ou experimentos práticos.

Afanasyev *et al.* [Afanasyev et al. 2013] também avaliam o ataque de inundação de interesse através de simulações, porém consideram diferentes cenários e uma rede de maior escala do que a usada no trabalho de Choi *et al.* Os autores também avaliam a contramedida baseada em um mecanismo de *push-back* proposta por Gasti *et al.*. Os resultados mostram que essa contramedida é eficiente, pois isola por completo os atacantes de modo que eles causem pouco ou nenhum impacto no desempenho percebido por usuários legítimos. Os autores não variam a posição do atacante. Para validar o mecanismo definiram o pior cenário, onde os interesses não serão satisfeitos com *cache* de um nó intermediário. Deste modo os dados são sempre encaminhados diretamente para o produtor. Os autores justificam essa medida afirmando que esse seria o pior caso para o ataque avaliado.

Os trabalhos a seguir tratam dos ataques de envenenamento de *cache*. Kim *et al.* [Kim et al. 2013b] investigam o impacto de fluxos de conteúdo de longa duração na CCN. A presença de fluxos de longa duração pode ter efeito similar ao ataque de envenenamento de *cache*. Se fluxos de longa duração ocuparem temporariamente um *cache* de um nó por determinado conteúdo, eles podem expulsar pedaços de conteúdos populares do *cache*. Consequentemente, reduz-se a taxa de acertos do *cache* (*cache hit*). Os resultados das simulações mostram que há degradação da taxa de acertos do *cache* quanto maior é o número de fluxos de longa duração.

Xie *et al.* [Xie et al. 2012] propõem um mecanismo chamado CacheShield, que usa dados estatísticos para verificar se um conteúdo é poluído ou não. O objetivo do CacheShield é aumentar a robustez do *cache* contra ataques de poluição, em particular perturbações de localidade (*locality disruption*). O mecanismo tenta impedir que conteúdos não populares sejam armazenados no *cache*. Quando um roteador habilitado para o CacheShield recebe um pedaço de conteúdo, o nó executa uma função de proteção que determina se o pedaço de conteúdo deve ser armazenado em *cache* ou não. Se a função de proteção retornar verdadeiro, o roteador encaminha o interesse e armazena em *cache* o pedaço de conteúdo. Se a função de proteção retorna falso, apenas o nome do pedaço de conteúdo (ou o *hash* de seu nome) e um contador são armazenados no *cache* num espaço

reservado, fora da área do CS. Se o mesmo pedaço de conteúdo, que ainda não está armazenado em *cache*, é solicitado novamente, o contador correspondente é incrementado, e a função de proteção é novamente executada. Assim a cada vez que o pedaço de conteúdo é avaliado pela função aumenta-se a chance desse ser efetivamente armazenado no *cache*. A função de proteção executa uma escolha aleatória com base em uma função logística: $\psi(t) = \frac{1}{1+e^{(p-t)/q}}$, onde t representa a *t-ésima* requisição para um determinado pedaço de conteúdo no CS, e p e q são os parâmetros da função. O mecanismo CacheShield será usado, para fins de comparação, com o mecanismo proposto por esse trabalho.

4. O Mecanismo Cache nFace

A contramedida proposta, chamada de Cache nFace, não tenta identificar o tráfego malicioso. A ideia principal da proposta é criar sub-*caches* em um nó a partir da divisão do espaço total de armazenamento em *cache* desse nó. Cada sub-*cache* está associado a uma das interfaces de rede do nó, e o seu tamanho é definido de acordo com a taxa de transmissão nominal da interface, como mostra o Algoritmo 1. Esse algoritmo recebe como entradas o tamanho total do *cache* do nó, S, o número de interfaces do nó, N, I o conjunto composto por todas as interfaces, e a taxa de transmissão q_i de cada interface i do nó e retorna o tamanho do sub-*cache* s_i de cada interface i. Q é soma das taxas de transmissão das N interfaces. Dessa forma, o tamanho de cada sub-*cache* é proporcional à capacidade de transmissão de cada interface.

Algoritmo 1: MECANISMO DE DIVISÃO DO CACHE NFACE

Entrada: S, q_i, N, I
Saída: Sub-Cache s_i
início
  $Q \leftarrow \sum_{n=1}^{N} q_n$
  para *cada* $i \in I$ **faça**
    $s_i \leftarrow \frac{S * q_i}{Q}$
  fim
fim
retorna s_i

Com o Cache nFace, somente o CS de um nó é modificado, ou seja, ao invés de um único *cache* agora existe um sub-CS para cada interface. Tanto a FIB quanto a PIT não são alteradas. Além disso, o encaminhamento de pacotes de interesse e de pacotes de dados não se alteram. Durante o encaminhamento de um pacote de interesse, todos os sub-*caches* são verificados para saber se o nó possui uma cópia do pedaço de conteúdo solicitado e não somente o sub-*cache* da interface de recepção do interesse. Os conteúdos continuam sendo encaminhados pelo caminho reverso ao percorrido pelo interesse. A única modificação é que, quando um interesse é atendido por um pacote de dados, o pedaço de conteúdo contido nesse pacote é armazenado no sub-*cache* correspondente à interface que recebeu o interesse. Se um determinado interesse está pendente por mais de uma interface de rede, o pacote de dados ficará armazenado no sub-*cache* correspondente a interface que primeiro recebeu o pacote de interesse.

Vale ressaltar também que um mesmo conteúdo não é armazenado em diferentes sub-*caches*. Para responder um interesse, é verificado se os outros sub-*caches* já possuem

o conteúdo. Portanto, não é encaminhado um novo interesse para um conteúdo que já está em um dos outros sub-*caches*. Assim, impede-se a duplicação de conteúdos no *cache* de um nó. Esse mesmo motivo obriga todos os sub-*caches* a usarem a mesma política de descarte definida para o CS. A premissa para o bom funcionamento do Cache nFace é que a probabilidade de pacotes de interesse usados em um ataque de conluio produtor-consumidor serem recebidos a altas taxas por todas as interfaces de um nó é pequena. Os roteadores de borda geralmente são mais afetados por estarem ligados a uma interface que pertence à rede do atacante. Assim, eles recebem altas taxas de interesses pela interface ligada à rede do atacante. Porém, a probabilidade de que as demais interfaces de rede do roteador de borda pertençam a redes de outros atacantes e/ou recebam tráfego malicioso encaminhado por outros roteadores em altas taxas é pequena. Deste modo, pelo menos uma interface que não pertence à rede do atacante pode responder a interesses legítimos que estão armazenados no seu sub-*cache*. Isso pode acontecer em cenários em que produtores e consumidores maliciosos tenham poucos saltos entre eles e a topologia não possua muitos caminhos alternativos entre eles. Isso indica que posição do atacante dentro da topologia e a quantidade de caminhos entre eles é fundamental para a eficiência do ataque.

5. Resultados

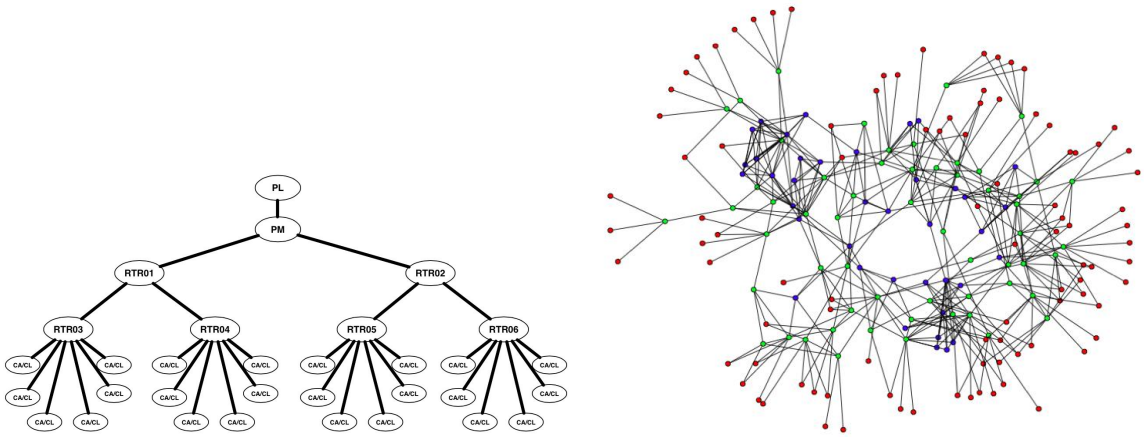
Nessa seção serão discutidas as avaliações dos mecanismos Cache nFace e CacheShield em diferentes cenários. Uma análise dos resultados é realizada para discutir a eficiência de ambos mecanismos sob ataque em conluio. Ainda são definidas as configurações, métricas e topologias utilizadas nas simulações.

5.1. Cenários de Avaliação

As topologias de rede usadas na avaliação do ataque em conluio e das contramedidas, são as seguintes. Uma topologia composta por 32 nós dispostos em forma de árvore, como mostra a Figura 1(a). Outra topologia em malha, criada com o mapeador de topologias Rocketfuel [Spring et al. 2004], com 192 nós, como mostra a Figura 1(b).

Na topologia em árvore, os 24 nós-folha são consumidores. O número de consumidores legítimos (CL) é fixo em todas as configurações e igual a 16. O número de consumidores maliciosos (CA) varia de 0 a 8. A posição dos CLs e dos CAs é definida aleatoriamente em cada rodada de simulação. O produtor legítimo (PL) é sempre o nó raiz. O produtor malicioso (PA) é o nó filho do nó raiz. Os demais 6 nós que compõem a topologia são os roteadores da rede (RTR). Os enlaces que interconectam os nós possuem taxa de transmissão de 100 Mb/s e atraso de 1 ms.

Na topologia em malha, tem-se 92 nós-folha como consumidores. O número de consumidores legítimos é fixo em todas as configurações e igual a 68. O número de consumidores maliciosos varia de 0 a 24, valores escolhidos para manter a mesma proporção da topologia em árvore. A posição dos consumidores legítimos e consumidores maliciosos é definida, aleatoriamente, em cada rodada de simulação. Existem 2 produtores legítimos e 1 produtor malicioso com posições aleatórias na rede. Os demais 97 nós que compõem a topologia são os roteadores (58 de borda e 39 de núcleo). Os enlaces que interconectam os nós possuem taxa de transmissão entre 124 kb/s e 92 Mb/s e atraso variante entre 1 e 70 ms.



(a) 32 nós. (b) 192 nós.

Figura 1. Topologias utilizadas nas simulações.

Os conteúdos são solicitados da seguinte forma. Os consumidores maliciosos enviam interesses para 12 conteúdos que são disponibilizados pelo produtor malicioso a taxas de 10, 100 e 500 interesses por segundo. Como cada pacote, seja de dados ou de interesse, tem tamanho máximo de 65.563 bytes [Kim et al. 2013a, Gallo et al. 2015], cada consumidor malicioso, no pior caso, vai enviar interesses a uma taxa de 65 kbps, 6,5 Mbps e 32,5 Mps respectivamente. Cada conteúdo malicioso possui 100 pedaços (*chunks*) e prefixos de nome diferentes. Os pedaços de conteúdos maliciosos são solicitados de acordo com o consumo sequencial, no qual um consumidor envia pacotes de interesse ordenados e de forma cíclica. Os consumidores legítimos sempre enviam 10 interesses/s para outros 12 conteúdos disponibilizados pelo(os) produtor(es) legítimo(s). Cada conteúdo legítimo possui 100 pedaços (*chunks*) e prefixos de nome diferentes. Os pedaços legítimos são solicitados seguindo uma distribuição Zipf com parâmetro $\alpha = 0,7$ [Breslau et al. 1999].

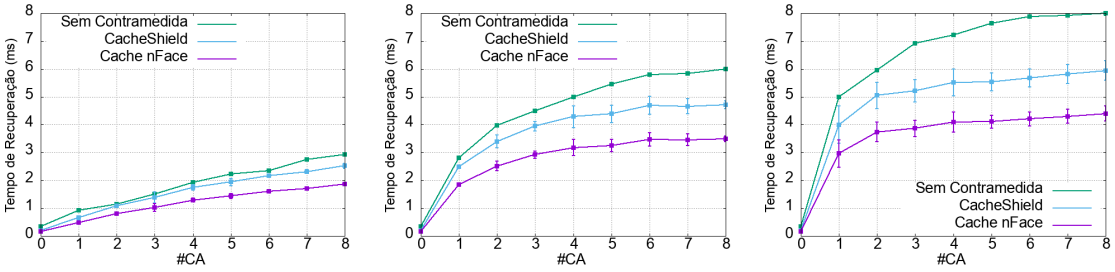
O *cache* dos consumidores legítimos e dos roteadores tem capacidade para armazenar até 1000 pedaços de conteúdo, e cada pedaço possui 1024 bytes. Os consumidores maliciosos não possuem *cache* para potencializar o ataque, isto é, sempre enviam interesses, independentemente se já receberam o conteúdo anteriormente ou não. A PIT tem tamanho ilimitado para que seja possível avaliar apenas o efeito do aumento da ocupação maliciosa no *cache* dos nós. A política de substituição de *cache* é a *Least Recently Used* (LRU) para todos os sub-*caches* do Cache nFace e para o CacheShield. Os parâmetros da função logística usada pelo CacheShield são $p = 20$ e $q = 1$. De acordo com os autores, esses valores são ideais para as topologias avaliadas [Xie et al. 2012].

O módulo ndnSIM [Afanasyev et al. 2012] do simulador NS-3 é usado na avaliação. Para cada configuração, são realizadas 50 rodadas de simulação, cada uma com duração de 180 s. Para os pontos dos gráficos obtidos, são calculados intervalos de confiança representados por barras verticais para um nível de confiabilidade de 95%.

5.2. Tempo Médio de Recuperação de Conteúdos Legítimos

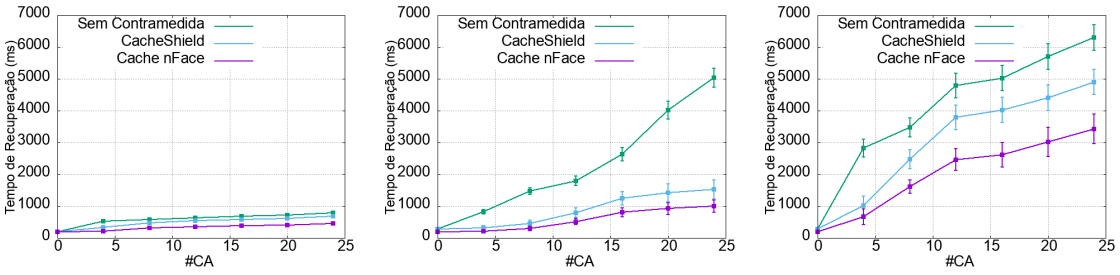
As Figuras 2 e 3 mostram o comportamento do tempo médio de recuperação de conteúdos legítimos em função do número de consumidores maliciosos, sendo que as Figuras 2(a), 2(b) e 2(c) referem-se à topologia em árvore, enquanto as Figuras 3(a), 3(b) e

3(c) referem-se à topologia em malha. O comportamento observado é o mesmo: quanto mais consumidores maliciosos, maior o tempo médio de recuperação de conteúdos. Da mesma forma, quanto maior a taxa de interesses maliciosos, maior o tempo médio de recuperação de conteúdos legítimos. Para topologia em árvore observa-se que o mecanismo CacheShield (Figura 2(c)), no pior caso em que atacantes enviam interesses a 500 interesses/s, é capaz de reduzir o tempo de recuperação de conteúdos em 2 ms quando comparado com a rede sob-ataque e sem uso de nenhuma contramedida. Isso representa uma redução de 25% na eficiência do ataque. Observa-se que o mecanismo Cache nFace, por sua vez, reduz em quase 50% a eficiência do ataque, se comparado com a rede sem contamedida, e em cerca de 35%, quando atacantes operam a 500 interesses/s, em relação ao CacheShield. É importante ressaltar que como os consumidores legítimos possuem *cache* e como eles sempre consomem de acordo com a popularidade, pode-se observar que sem ataque o consumo é, muitas vezes, feito do próprio *cache* do nó, o que resulta em um tempo de recuperação inferior a 1 ms.



(a) 10 interesses/s. (b) 100 interesses/s. (c) 500 interesses/s.

Figura 2. O tempo de recuperação de conteúdos legítimos na topologia em árvore.



(a) 10 interesses/s. (b) 100 interesses/s. (c) 500 interesses/s.

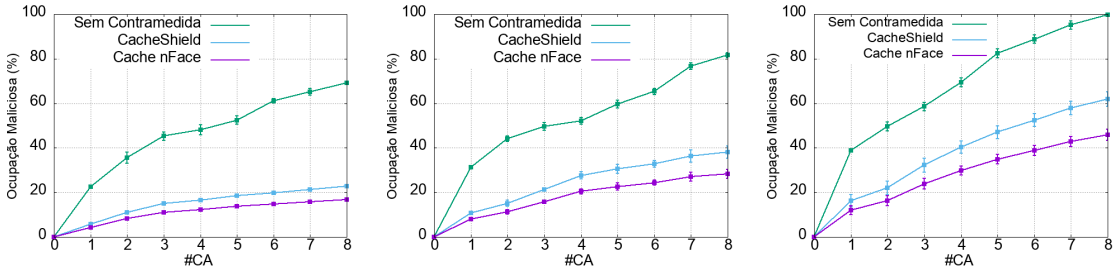
Figura 3. O tempo de recuperação de conteúdos legítimos na topologia em malha.

A efetividade do ataque pode ser observada também na topologia em malha. Quando não há contramedida na rede, Figuras 3(a), 3(b), 3(c), tem-se um tempo médio de recuperação de conteúdos 2 vezes maior quando tem-se 24 consumidores maliciosos enviando 10 interesses/s, 12 vezes maior quando enviam 100 interesses/s e 16 vezes maior quando operam a 500 interesses/s, se comparado com o Cache nFace em mesmas condições. Outro resultado interessante para topologia em malha, quando atacantes operam a 500 interesses/s, é que o mecanismo Cache nFace (Figura 3(c)) é cerca de 35% mais eficaz que o CacheShield, e ainda reduz, em cerca de 50%, a eficiência do ataque quando comparado com a rede sem contramedida.

5.3. Ocupação Maliciosa dos Caches e Taxa de Erro do Cache

O motivo da contramedida Cache nFace ser mais eficiente é que ela reduz a ocupação maliciosa dos *caches* dos roteadores e, consequentemente, reduz a taxa de erro de *cache* dos conteúdos legítimos se comparada ao CacheShield. Esses resultados são mostrados pelas Figuras 4, 5, 6 e 7. Na topologia em árvore, enquanto tem-se na curva sem ataque da Figura 2(b) o tempo de recuperação de 6 ms para taxa de 100 interesses/s com 8 consumidores maliciosos, tem-se ocupação maliciosa e taxa de erro do *cache* na ordem de 80% (Figura 6(b)). Essa característica implica que, sob ataque, os consumidores devem dar mais saltos para recuperar o conteúdo solicitado. Como, na topologia em árvore, o atraso de cada enlace é de 1 ms, conclui-se que os conteúdos legítimos são recuperados mais frequentemente de nós que estão a mais saltos do consumidor do que os nós de borda. Isso indica que os roteadores de borda estão com uma alta ocupação de conteúdos maliciosos em seu *cache*.

Os mecanismos avaliados reduzem para, aproximadamente 5 ms com o CacheShield e 3 ms com o Cache nFace, com 8 consumidores maliciosos enviando 100 interesses/s. Isso indica que com CacheShield, na topologia em árvore, faz-se em média 3 saltos para recuperar o conteúdo, enquanto com Cache nFace são necessários em média 2 saltos. Quando a rede está sob ataque e não utiliza nenhum dos mecanismos estudados, operando na mesma taxa e quantidade de consumidores do exemplo anterior, tem-se em média 4 saltos. Deste modo, pode-se dizer que o Cache nFace reduz pela metade o tempo de recuperação de conteúdos legítimos nesse cenário, reduzindo a eficiência do ataque.

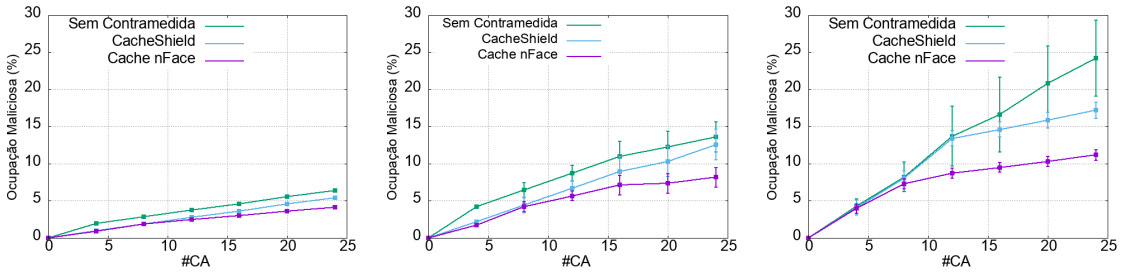


(a) 10 interesses/s. (b) 100 interesses/s. (c) 500 interesses/s.

Figura 4. O percentual de ocupação do *cache* dos roteadores por conteúdos maliciosos para as contramedidas avaliadas na topologia em árvore.

Na topologia em malha, Figuras 5(a), 5(b) e 5(c), observa-se uma ocupação maliciosa de cerca de 20% quando existem na rede 20 consumidores maliciosos enviando 500 interesses/s. Portanto, na topologia em malha, se cerca de 10% da rede é composta de consumidores maliciosos, a ocupação maliciosa dos *caches* do roteadores de conteúdo já compromete o consumo de conteúdo legítimo. Por outro lado, o mecanismo CacheShield pode reduzir em cerca de 5 pontos percentuais e o mecanismo Cache nFace em 10 pontos percentuais essa ocupação maliciosa, como se observa na Figura 5(b).

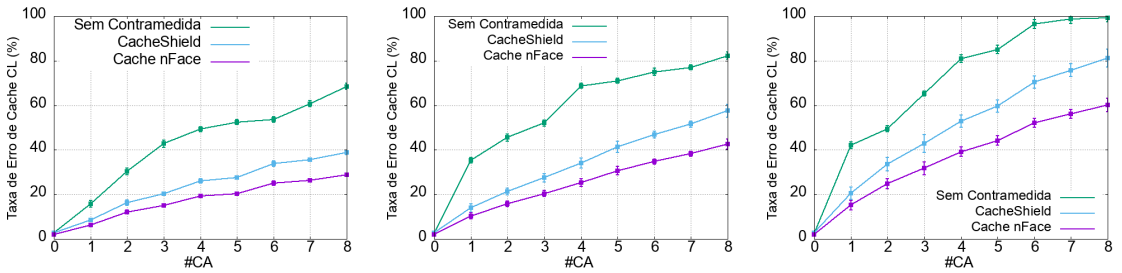
Na topologia de em malha, observa-se uma ocupação maliciosa e taxa de erro do *cache* menor que na topologia em árvore. Isso deve-se ao fato que a topologia em árvore tem a característica de ter o produtor malicioso no caminho do produtor legítimo, obrigando, assim, um consumidor legítimo a utilizar o *cache* de, pelo menos, um nó comprometido para recuperar um conteúdo legítimo. Ainda é possível observar uma maior



(a) 10 interesses/s. (b) 100 interesses/s. (c) 500 interesses/s.

Figura 5. O percentual de ocupação do *cache* dos roteadores por conteúdos maliciosos para as contramedidas avaliadas na topologia em malha.

eficiência do ataque em comprometer o consumo de conteúdos legítimos quando se tem 24 consumidores maliciosos enviando 500 interesses/s. Nessa curva, observa-se uma elevada ocupação maliciosa e taxa de erro do *cache*, como mostrado nas Figuras 5(c) e 7(c) em ordens de 25% e 42% respectivamente. Pode-se concluir que com 24 consumidores maliciosos, que representam 12,65% do total de nós, pode-se aumentar a taxa de erro do *cache*, comprometendo o consumo de conteúdos legítimos. Os mecanismos estudados, porém, reduzem a eficiência do ataque. O CacheShield pode, por exemplo, reduzir a taxa de erro do cache (Figura 7(b)) para 100 interesses/s com 15 consumidores maliciosos em cerca de 40%, enquanto o Cache nFace, na pior configuração (500 interesses/s para 24 consumidores maliciosos) pode reduzir a ocupação maliciosa (Figura 5(c)) também em cerca de 40%. Ou seja, em taxas de 500 interesses/s com 24 consumidores maliciosos, o mecanismo Cache nFace tem a mesma eficiência que o CacheShield em taxa inferior de 100 interesses/s com 24 consumidores maliciosos, se mostrando mais eficaz para reduzir os efeitos do ataque nos cenários avaliados.

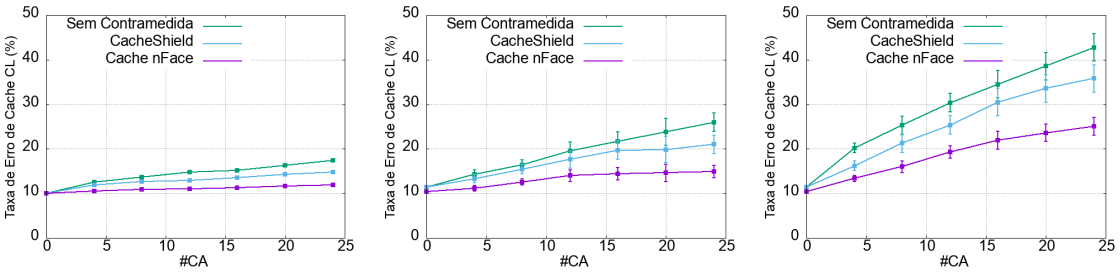


(a) 10 interesses/s. (b) 100 interesses/s. (c) 500 interesses/s.

Figura 6. A taxa de erros de *cache* para os conteúdos legítimos para as contramedidas avaliadas na topologia em árvore.

5.4. Conteúdos Recuperados do Produtor Legítimo

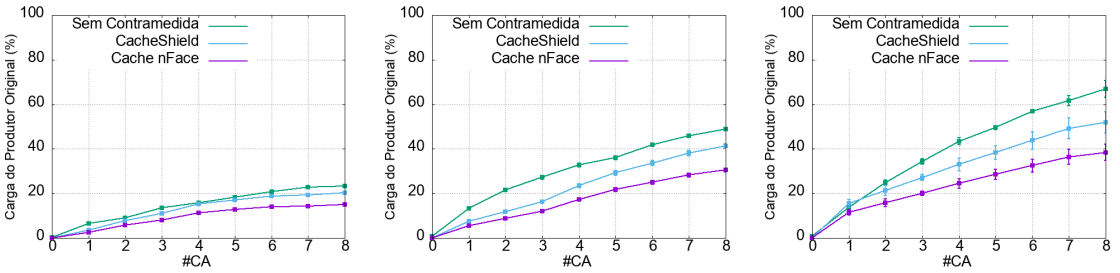
As Figuras 8 e 9 mostram o percentual de conteúdos legítimos recuperados do produtor legítimo em função do número de consumidores maliciosos e da taxa de envio de interesses por esses nós. Esses resultados corroboram que o ataque em conluio reduz e eficiência do emprego do *cache* pela CCN. A Figuras 8(a), 8(b) e 8(c) mostram que, se não há ataque, cerca de 0,5% dos conteúdos solicitados são recuperados diretamente do produtor original. Nesse caso, cada conteúdo legítimo é recuperado do produtor original



(a) 10 interesses/s. (b) 100 interesses/s. (c) 500 interesses/s.

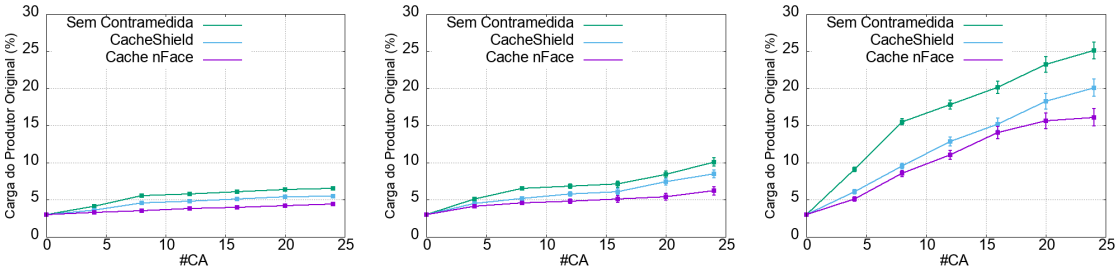
Figura 7. A taxa de erros de *cache* para os conteúdos legítimos para as contramedidas avaliadas na topologia em malha.

no máximo duas vezes, até que seja armazenado pelos nós RTR1 e RTR2. Porém, basta se ter 4 consumidores maliciosos enviando 10 interesses/s (Figura 8(a)) para que esse valor aumente para cerca de 12%. No pior caso, os consumidores legítimos estão recuperando cerca de 67% dos conteúdos legítimos diretamente do produtor legítimo na topologia em árvore. Com os mecanismos Cacheshield e Cache nFace tem-se redução de no carga no produtor legítimo. Isso deve-se ao fato de que roteadores de conteúdo estarem com menos ocupação maliciosa, fazendo aumentar a taxa de acerto do *cache* e recuperando conteúdos legítimos em menos saltos.



(a) 10 interesses/s. (b) 100 interesses/s. (c) 500 interesses/s.

Figura 8. Percentual de carga do produtor para as contramedidas avaliadas na topologia em árvore.



(a) 10 interesses/s. (b) 100 interesses/s. (c) 500 interesses/s.

Figura 9. Percentual de carga do produtor para as contramedidas avaliadas na topologia em malha.

É interessante observar que, sob ataque, apenas 4 atacantes, enviando 100 interesses/s, elevam a carga do produtor a cerca de 38% (Figura 8(b)), com CacheShield nas

mesmas configurações tem-se apenas 22%. Para o mecanismo Cache nFace, nas mesmas configurações, observa-se que os mesmos 4 atacantes deixam a carga no produtor legítimo em 19%. O mecanismo Cache nFace mostra-se mais eficiente em reduzir o tempo de recuperação, a ocupação maliciosa, a taxa de erro do *cache* e a carga no produtor que o CacheShield. As características observadas na topologia em árvore podem ser observadas também na topologia me malha para o consumo direto do produtor legítimo, Figuras 9(a), 9(b) e 9(c). Na topologia em malha, ainda tem-se no pior caso, no qual 24 consumidores maliciosos enviam 500 interesses/s, uma carga do produtor legítimo de cerca de 25% sem o uso das contramedidas. Assim, como os pacotes de dados são recuperados diretamente do produtor, a eficiência da rede em entregar conteúdos legítimos mais próximos do consumidor é comprometida. Observa-se, ainda, que o mecanismo CacheShield reduz a carga produtor em cerca de 20%, enquanto o mecanismo Cache nFace reduz a carga do produtor em cerca de 45% nas configurações avaliadas.

Uma característica relevante para a avaliação é que o Cache nFace divide o CS do nó em sub-*caches*. Essa característica é importante porque se um determinado nó, que conta com quatro interfaces de rede, por exemplo, usar o Cache nFace, terá o CS divido em quatro sub-*caches*, que tomam decisões sobre armazenar um conteúdo de forma independente. Isso dificulta a ação do atacante, pois seria necessário afetar todos os quatro sub-*caches* para o nó negar serviço à rede. O funcionamento do Cache nFace impede que um sub-*cache* seja poluído por outro do mesmo nó. Nas topologias estudadas, o mecanismo Cache nFace se mostrou mais eficaz que o CacheShield em todos os cenários avaliados.

6. Conclusão

Nesse trabalho foi proposta uma contramedida para o ataque em conluio produtor-consumidor chamada Cache nFace. O funcionamento do mecanismo baseia-se em criar sub-*caches* em um nó, a partir da divisão do espaço total de armazenamento em *cache* desse nó. Cada sub-*cache* está associado a uma das interfaces de rede do nó.

A contramedida foi comparada com outra encontrada na literatura através de simulação para diferentes topologias e configurações de ataque. Resultados mostraram que a contramedida Cache nFace reduz em cerca de 50% a efetividade do ataque e se mostra mais eficaz nos cenários avaliados que o CacheShield. A contramedida Cache nFace é capaz de comedir o ataque nas configurações avaliadas, reduzindo sua eficiência na topologia malha e na topologia em árvore. Como trabalhos futuros pretende-se avaliar o efeito de aumentar o número de produtores maliciosos para aumentar a probabilidade de ter mais do que uma interface de rede sendo atacada. Ou, ainda, avaliar quantos sub-*caches* de um nó possuem conteúdos maliciosos.

Agradecimentos

Este trabalho é apoiado por Dinter UFF/UFAC, CNPq, CAPES, FAPERJ, Proppi/UFF, TBE/ANEEL e CELESC/ANEEL.

Referências

Afanasyev, A., Mahadevan, P., Moiseenko, I., Uzun, E., and Zhang, L. (2013). Interest flooding attack and countermeasures in named data networking. In *IFIP Networking*, pages 1–9.

Afanasyev, A., Moiseenko, I., and Zhang, L. (2012). ndnSIM: NDN simulator for NS-3. Technical Report NDN-0005, NDN.

Breslau, L., Cao, P., Fan, L., Phillips, G., and Shenker, S. (1999). Web caching and zipf-like distributions: evidence and implications. In *IEEE Conference on Computer Communications - INFOCOM*, pages 126–134.

Brito, G. M., Velloso, P. B., and Moraes, I. M. (2012). Redes orientadas a conteúdo: Um novo paradigma para a Internet. In *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC*, pages 211–264.

Brito, G. M., Velloso, P. B., and Moraes, I. M. (2013). *Information-Centric Networks, A New Paradigm for the Internet*. FOCUS - Networks and Telecommunications Series. Wiley-ISTE, 1 edition.

Gallo, M., Perino, D., and Muscariello, L. (2015). Content-centric networking packet header format. Technical Report BCP-78, Internet Engineering Task Force.

Gasti, P., Tsudik, G., Uzun, E., and Zhang, L. (2013). DoS and DDoS in named-data networking. In *International Conference on Computer Communications and Networks - ICCCN*, pages 1–7.

Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and Braynard, R. (2009). Networking named content. In *International Conference on emerging Networking EXperiments and Technologies - CoNEXT*.

Kim, J., Shin, D., and Ko, Y.-B. (2013a). Top-ccn: Topology aware content centric networking for mobile ad hoc networks. In *2013 19th IEEE International Conference on Networks (ICON)*, pages 1–6.

Kim, Y., Kim, U., and Yeoml, I. (2013b). The impact of large flows in content centric networks. In *IEEE International Conference on Network Protocols - ICNP*, pages 1–2.

Nasserala, A. and Moraes, I. M. (2015). Artigo 1. In *XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 641–654.

Nasserala, A. and Moraes, I. M. (2016a). Artigo 2. *IEEE Latin America Transactions*, 14(6):3003–3010.

Nasserala, A. and Moraes, I. M. (2016b). Artigo 3. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 849–852.

Smetters, D. and Jacobson, V. (2009). Securing network content. Technical Report TR-2009-1, Xerox Palo Alto Research Center - PARC.

Spring, N., Mahajan, R., Wetherall, D., and Anderson, T. (2004). Measuring ISP topologies with Rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16.

Xie, M., Widjaja, I., and Wang, H. (2012). Enhancing cache robustness for content-centric networking. In *IEEE Conference on Computer Communications - INFOCOM*, pages 2426–2434.

