



Research article

Optimal filter assignment policy against link flooding attack

Rajorshi Biswas^{a,*}, Jie Wu^b, Wei Chang^c, Pouya Ostovari^d^a Information Sciences and Technology, Penn State University (Berks), Reading 19610, USA^b Department of Computer and Information Sciences, Temple University, Philadelphia 19122, USA^c Department of Computer Science, Saint Joseph's University, Philadelphia 19131, USA^d KLA Corporation, Milpitas 95035, USA

ARTICLE INFO

Article history:

Received 2 December 2023

Revised 13 January 2024

Accepted 19 March 2024

Available online 15 April 2024

Keywords:

Botnet

DDoS defense

Quality-of-service

Filter router

Link flooding attack

Network security

ABSTRACT

A Link Flooding Attack (LFA) is a special type of Denial-of-Service (DoS) attack in which the attacker sends out a huge number of requests to exhaust the capacity of a link on the path the traffic comes to a server. As a result, user traffic cannot reach the server. As a result, DoS and degradation of Quality-of-Service (QoS) occur. Because the attack traffic does not go to the victim, protecting the legitimate traffic alone is hard for the victim. The victim can protect its legitimate traffic by using a special type of router called filter router (FR). An FR can receive server filters and apply them to block a link incident to it. An FR probabilistically appends its own IP address to packets it forwards, and the victim uses that information to discover the traffic topology. By analyzing traffic rates and paths, the victim identifies some links that may be congested. The victim needs to select some of these possible congested links (PCLs) and send a filter to the corresponding FR so that legitimate traffic avoids congested paths. In this paper, we formulate two optimization problems for blocking the least number of PCLs so that the legitimate traffic goes through a non-congested path. We consider the scenario where every user has at least one non-congested shortest path in the first problem. We extend the first problem to a scenario where there are some users whose shortest paths are all congested. We transform the original problem to the vertex separation problem to find the links to block. We use a custom-built Java multi-threaded simulator and conduct extensive simulations to support our solutions.

© 2024 The Author(s). Published by Elsevier B.V. on behalf of Shandong University. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

1. Introduction

One of the greatest threats to critical services on the Internet is distributed denial-of-service (DDoS) attacks. The objective of a DDoS attack is to generate a lot of packets from different sources to exhaust the bandwidth or resources used by a victim. A coordinator sends commands to its workers who continuously send requests to the victim. The workers are known as bots and the network of workers is known as a botnet. Bots are usually user computers that are infected with malicious programs. Because of the huge amount of traffic or service requests a machine or network resource becomes unavailable to its users.

DDoS attacks have evolved a lot during the last decade. The cost of conducting a DDoS attack has also decreased. For example, renting 1000 bots from any location costs from a few dollars to a little more than 100 USD [1]. Attackers' strategies, capabilities, and goals have also evolved a lot during these years. For example, an attack against Spamhaus in 2013 revealed that the attacker can change the target adaptively from the endpoint server according to the defense mechanism. Another attack against ProtonMail in

2015 demonstrated that the attackers changed their strategy in real time according to the defense mechanism [2]. The moving target and attack strategy form an interactive game scenario between the defender and the attacker.

The link flooding attack is one of the most challenging DDoS attacks because of its slightly different strategy and objective. The packets generated by the bots are destined to the victim in the traditional DDoS attack. In a link flooding attack, the traffic is not destined to the victim but they congest at least one of the links on the paths from the legitimate users to the victim. The legitimate traffic suffers from packet drop and delay because of the congested links. In the worst case, this attack can disconnect a server from the Internet.

Fig. 1 shows a scenario of a link flooding attack. The bots attached to NAT₁ generate huge traffic towards the decoy server v'_2 . Decoy server servers are used to receive the attack packets from bots. decoy server servers may be owned by the attacker and the attack packets congest one or more links on the path to decoy servers. The packets follow the path $r_1 \leftrightarrow r_6 \leftrightarrow r_3 \leftrightarrow r_4$. Similarly, the traffic from the bots attached to NAT₃ follows the path $r_5 \leftrightarrow r_4 \leftrightarrow r_3$ to reach v'_1 . The link $r_3 \leftrightarrow r_4$ becomes congested because of the traffic from bots from NAT₁ and NAT₂. When the legitimate packets from the user attached to NAT₂

* Corresponding author.

E-mail address: raajorshi@psu.edu (R. Biswas).

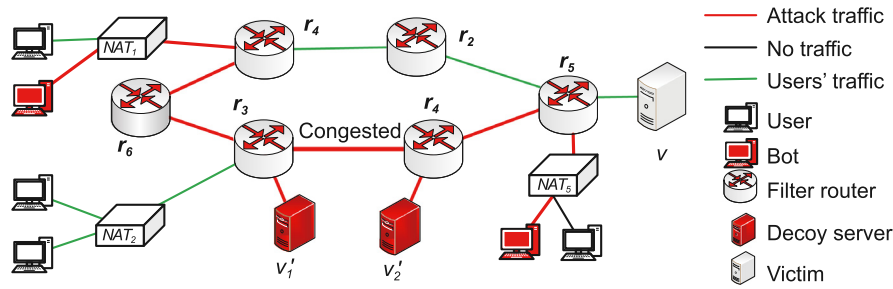


Fig. 1. Link flooding attack by bots ($r_1 - r_6$ are filter routers; v'_1 and v'_2 are decoy servers; v is the victim; link $r_3 \leftrightarrow r_4$ is congested).

travel the shortest path $r_3 \leftrightarrow r_4 \leftrightarrow r_5$, they face delay, packet drop, and DoS. On the other hand, legitimate traffic from users attached to NAT_1 and NAT_3 do not encounter any congested links and the victim observes a good traffic rate from them. Because of low data rates from users attached to NAT_3 , the victim assumes that either any or both of the $r_3 \leftrightarrow r_4$ and $r_4 \leftrightarrow r_5$ links are congested. The victim can disable a link for the packets destined to him by sending a filter to the corresponding filter router (FR). The FRs are special kinds of routers. In addition to the routing task, they do two operations: packet marking and traffic filtering. The packet marking operation probabilistically appends FR's IP address to the packets it forwards. The marked packets are used to identify paths traveled by a users' packets. The victim can obtain traffic topology by combining the paths from all sources. Traffic filtering is associated with two operations: receiving filters from victims and applying the filters to block or modify the routes that packets follow to the victim. The concept of FR is not new and well studied in [3–5] for preventing DDoS attacks, but it has never been studied for link flooding attacks.

Assigning a filter to an FR has a cost. The costs include monetary and communications overhead. For example, if the victim and FR belong to different ISPs, then the ISP of FR can earn by accepting filters from the victim. On the other hand, the victim improves its QoS by changing the user's route by deploying filters. This way the victim and owner ISP of FR both get benefited. Therefore, for the victim, it costs a lot to disable all the PCLs. In this paper, we focus on directing all legitimate traffic through non-congested paths with a minimum number of filters. The main contributions of our paper are:

- (1) We formulate an optimization problem for directing legitimate traffic through the alternate shortest non-congested paths with the minimum number of filters/blockage.
- (2) We propose an optimal solution in polynomial time by narrowing down the problem to the vertex separation problem.
- (3) We formulate another optimization problem for the scenario where there is no alternate non-congested shortest path from the users. We solve the problem by narrowing down the problem to the first problem.
- (4) We provide extensive simulation with synthetic and real topology and compare it with existing approaches to validate our model.

The remainder of this paper is arranged as follows: Section 2 presents some related works and their limitations. In Section 3, we present the system model for preventing link flooding attacks. In Sections 4 and 5, we formulate a problem in a special scenario where there is a non-congested shortest path from every user, and we provide a solution. In Sections 6 and 7, we extend the problem to a more general scenario and present an optimal solution. In Section 8, we present the simulation results. Section 9 describes some limitations and future works.

2. Related works

There exist many works defending link flooding attacks. Many statistical methods, including correlation, entropy, covariance, divergence, cross-correlation, and information gain have been used for network traffic analysis to detect anomalous traffic [6–8]. There are multi-path routing mechanisms that help to mitigate congestion, including [9–12]. These schemes do not give priority to legitimate traffic and the destination machine does not have any control over it.

Several methods employ routers to identify attack traffic and block them. The SPIFFY [13] logically increases link capacity when it detects congestion. After the capacity increases, the attacker may increase the data rate of each bot to keep the cost constant and be identified. The ColDef [14] enables routers to distinguish low-rate attack flows from legitimate flows. In this mechanism, the domains that are uncontaminated by the botnet help route the legitimate traffic. In [15], the authors propose PUSHBACK, a router functionality-based mechanism that enables each router to detect and preferentially drop packets that likely belong to an attacker. The upstream routers are also notified of the drop event so that they can use the resources for legitimate traffic. In [16], the authors propose a router subsystem called FLoc (Flow Localization) that provides a differential bandwidth guarantee to legitimate traffic at congested links. The FLoc distinguishes between the traffic from bot-contaminated and uncontaminated domains. Legitimate traffic from uncontaminated domains has more homogeneous congestion characteristics – including mean packet drop rates and mean delay – than flows of contaminated domains. Based on the traffic characteristics, FLoc determines how much bandwidth to allocate to the traffic from a domain.

Authors in [17] presented a defense framework that Markov model-based reinforcement agents. It learns attack strategies and actions and dynamically selects the optimal defense mechanism which is a combination of filtering, limiting, and rerouting.

In [18], authors present network policies and security validation mechanisms to enhance understanding of SDN behavior and develop strategies for securing SDN-based networks. The proposed model collects traffic data and applies AI algorithms for detecting various types of network flooding attacks. The framework deploys the best-performing model in the controller.

In [19], authors proposed a scheme that mitigates link flooding attacks by using some BGP rules in BGP routers. When link congestion is detected, the BGP router advertises its neighbors in such a way so that the congested link is avoided. When a BGP router detects congestion, it creates a false path by appending its address and advertises the false path. Other BGP routers from different autonomous systems find that the path has a loop and avoid it.

We discussed two types of existing defense systems: (1) attack packet/flow detection at a router followed by packet drop, rate control, and redirection and (2) identification of congested links at a router or controller and avoiding the link. The first type

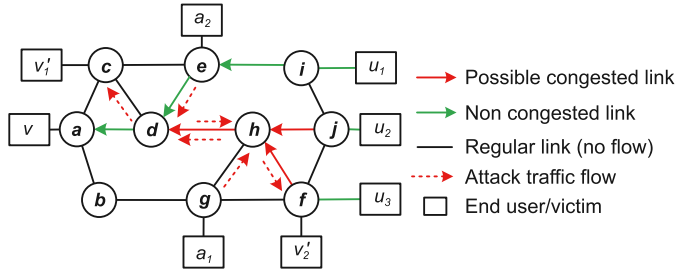


Fig. 2. System model (a_1 and a_2 are attackers; v_1' and v_2' are decoy servers; v is the victim; u_1, u_2 , and u_3 are users).

increases router computation overhead and the false-positive creates a great loss to the victims. Besides, bots are smart enough to create traffic similar to users and are unidentifiable using statistical techniques. For the second type, the identification of the congested link for a victim far from the congested link is tough without additional access. The victim can distinguish between the attacker and legitimate traffic. Therefore, a defense system in which routing is controlled by the victim is necessary.

The most related work preventing DDoS attacks is the probabilistic filter scheduling (PFS) system [3]. The victim generates filters and sends them to the upstream FRs. The FRs send the filters to their upstream FRs and thus the filters propagate to the effective FRs. An adaptive version of PFS is proposed in [4]. The system directly sends filters to the highly capable FRs first, then the filters propagate to the effective FRs. However, these two systems cannot optimally select the FRs when there is a limitation on selecting FRs. It is not applicable for link flooding attack protection.

3. System model

In this section, we define the network and the attack model.

3.1. Network model

Our network is composed of network address translators (NATs), filter routers (FRs), bots, users, and a victim. We assume that all the users are connected to FRs through NAT. We also assume that there is at least one non-congested path from every user to the victim.

The FRs are a special kind of router capable of two functions. Firstly, they can mark packets that is used to construct traffic topology for the victim. Secondly, they can receive filters from the victim and apply the filters to block a link. A filter is a special instruction to an FR to block an incident link. When the victim v sends a filter to an FR, any packet destined to the v will be affected by the filter. For example, Fig. 2 shows a network with a victim v , two decoy servers (v_1', v_2'), two attackers (a_1, a_2), three users (u_1, u_2, u_3), and several filter routers ($a-j$). The links with green and red arrows indicate the traffic flow directions of legitimate and attack traffic. Let us assume that a filter “block link $h \leftrightarrow d$ ” is sent to an FR h by v . The FR h may forward an incoming packet through link $h \leftrightarrow d$ if the destination is not v . Therefore, the attack traffic from a_1 heading to the decoy server v_1' is not blocked. If the packet is destined to v , then the FR will never forward the packet through the link $h \leftrightarrow d$. FR h will not forward user traffic from u_2 through $h \leftrightarrow d$.

The implementation of an FR is possible on both regular routers and software-defined networking switches. Some open-source router operating systems can be modified to adopt FR functionality. Nowadays, software-defined networking switches

have some functionalities that can be used to adapt the filtering mechanism. We have clarified these in the System Model section in this revision.

Sometimes an attacker can impersonate v and send filters to FRs but it is identifiable using a simple handshaking protocol. For example, an attacker may send a wrong filter by sending a filter request message as the victim. When a filter request comes to a victim, it will ask the sender whether it has sent the filter or not. The attacker will not get the verification message and will not succeed in the filter spoofing attack.

The system is a four-phase system. **In phase I**, the victim generates or obtains the topology and traffic topology (the way the traffic comes to the victim). The traffic topology can be obtained from packet marking by filter routers [20].

In phase II, the victim identifies the users that are suffering from congestion by using the packet arrival rate and finds some PCLs. It is hard for the victim to know which links are congested using the data rate of users. For example, in Fig. 2, the attackers congest the link $d \leftrightarrow h$. As a result, traffic from u_2 and u_3 suffer from congestion. The victim v observes that data rates from u_2 and u_3 are below the regular data rate. Therefore, one or multiple links on the path $a \leftrightarrow d \leftrightarrow h \leftrightarrow j$ and $a \leftrightarrow d \leftrightarrow h \leftrightarrow f$ may be congested. The victim also observes that the data rate from u_1 is good. Therefore, none of the links on $a \leftrightarrow d \leftrightarrow e \leftrightarrow i$ are congested. With this observation, the victim concludes that $d \leftrightarrow h, h \leftrightarrow j$, or $h \leftrightarrow f$ links have a possibility to be congested.

In phase III, After finding the PCLs, the victim redirects its incoming traffic to avoid the PCLs by sending some filters. A trivial solution is to block all PCLs by sending filters to the FRs that are associated with them. For example, if v wants to disable link $d \leftrightarrow h$, it would send a filter “block $d \leftrightarrow h$ ” to FR h . Sending a filter to FR has a cost. There are too many FR routers in the network, and the victim wants to reduce the filtering cost. Moreover, there is no need to block all of the PCLs. For example, if v blocks $h \leftrightarrow j$ and $f \leftrightarrow h$, then all legitimate traffic will follow non-congested paths. Therefore, blocking $d \leftrightarrow h$ is unnecessary. The victim wants legitimate traffic to follow the non-congested path by blocking a minimum number of links by spending the minimum amount of resources. The victim can run a centralized algorithm to find the minimum number of required blocking. The attack traffic continues to follow the same path because the filters do not affect any packet that is not destined for the victim. Therefore, the system does not reduce network congestion by blocking attack traffic. It only forces the legitimate traffic to follow a non-congested path. Whenever we mention blocking a link, we refer to blocking only the victim’s legitimate traffic through that link.

In phase IV, the FRs remove the unused filters from their storage. As the FRs have limited storage capacity for filters, it is necessary to reduce the used storage by removing the unused filters. Otherwise, the attacker can flood the FRs by sending useless filters and jamming the filter routers. The filters from the attackers will be evicted soon because they are most likely not being used.

3.2. Attack model

The attacker knows the topology and locations of the victim and users but does not know the traffic topology. The attacker selects one or multiple links as a target after analyzing the topology and the locations of the users and the victim. The target links are used by a reasonable number of users and they are reachable by the flows from bots to decoy servers. Then, based on the locations of the target links, the attacker selects decoy servers and bots to generate attack traffic. The selection of a pair $\langle bot, decoy server \rangle$ is done in such a way that the traffic from the

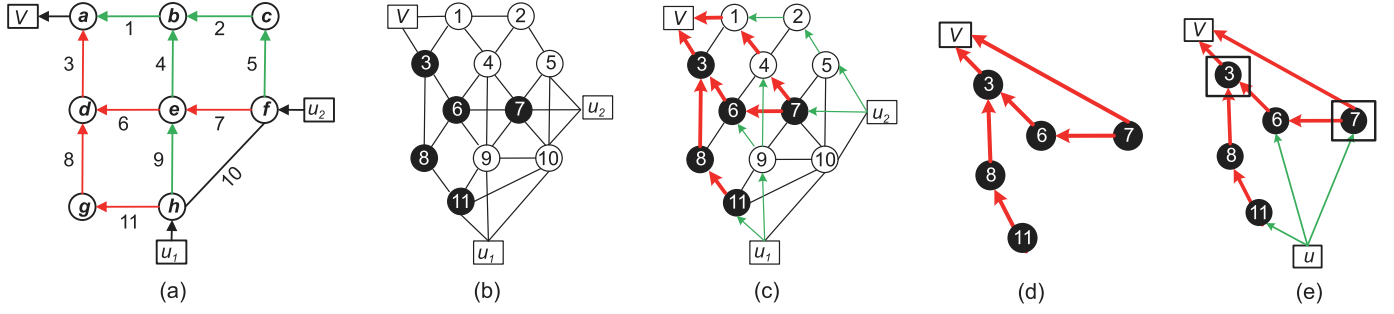


Fig. 3. Problem I example. (a) Original topology G . (b) Line graph G' generated from G . (c) Line graph G' with flows added. (d) G' with black nodes and connected links only. (e) Final graph G'' with filters.

Table 1

Table of notations.

Notations	Description
v	Victim
B	Number of blocked edges
P_u^s	Set of the shortest paths from u to v
P_u	Set of paths from u to v
$P_u^s(B)$	Set of shortest paths from u to v after blocking B links
$G = (V, E)$	Topology graph
$G' = (V', E')$	Line graph
$G'' = (V'', E'')$	Congested flow graph
U	Number of users
e_{ij}	Edge between node i and j
L_c	Set of congested links
$C_e(e)$	Color of edge e
$C_n(n)$	Color of node n
$C_f(f)$	Color of flow f
$Neighbor(n)$	Neighbor set of node n

bot to the decoy server passes through at least one targeted link. The traffic of several attack flows cannot congest a link, but when a large number of attack flows pass through the link it becomes congested. As a result, the legitimate traffic that passes through the congested links suffers from packet drop, delay, and low data rate.

In Fig. 2, after analyzing the topology, location of the victim, and the users, the attacker finds that link $a \leftrightarrow d$ carries the most legitimate traffic but cannot be reached by the decoy servers and bots. The second most important link is $d \leftrightarrow h$ which carries two legitimate flows and is reachable by bots and decoy servers. Therefore, the attacker starts $\langle a_1, v_1' \rangle$ and $\langle a_2, v_2' \rangle$ flows by sending commands to a_1 and a_2 . $\langle a_1, v_1' \rangle$ and $\langle a_2, v_2' \rangle$ pass through paths $g \leftrightarrow h \leftrightarrow d \leftrightarrow c$ and $e \leftrightarrow d \leftrightarrow h \leftrightarrow f$. The traffic created by a bot cannot congest a link so that $c \leftrightarrow d$, $e \leftrightarrow d$, $g \leftrightarrow h$, and $f \leftrightarrow h$ are not congested. Link $d \leftrightarrow h$ becomes congested because both flows pass through it. As a result, legitimate traffic from u_2 and u_3 suffer from low data rates.

4. Problem definition

In this section, we formulate the problem for finding a minimum number of links to block and provide an optimal solution [21]. The notations used in this paper are summarized in Table 1. To solve the problem, we break the problem into two steps. In the first step, we assume that there exist multiple users and at least one shortest non-congested path from every user (defined as Problem I). In the second step, we extend the problem to solve the situation where there exists some entry node from which all of the shortest paths are possibly congested (defined as Problem II).

Algorithm 1 Find a minimum number of blockage in G

Input: Topology G , PCLs L_c .

Output: A set of links to block from L_c .

```

1: Procedure: FIND-ASSIGNMENT( $G$ )
2:    $G' \leftarrow$  the line graph of  $G$ .
3:   CREATE-FLOW( $G'$ )
4:    $G'' \leftarrow$  ELIMINATE( $G'$ )
5:    $g \leftarrow$  minimum vertex separation set in  $G''$ 
6:   return  $g$ 

```

Problem I. Find the minimum number of filter assignments so that all legitimate traffics follow non-congested shortest paths.

In this problem, we assume that, from every user, there is at least one non-congested shortest path. Let, P_u^s be the set of the shortest paths from a user u to victim v . L_c is the set of PCLs. Some of the paths in P_u^s contain PCLs and are defined as possible congested paths. After deploying B number of filters to FRs in G , the set of the shortest paths from u is $P_u^s(B)$.

In this case, the problem can be expressed as the following optimization problem:

$$\begin{aligned}
 &\text{minimize } B \\
 &\text{subject to } \forall_{1 \leq u \leq U, p \in P_u^s(B)} \quad p \cap L_c = \emptyset, \\
 &\quad P_u^s(B) \subseteq P_u^s
 \end{aligned} \tag{1}$$

5. Optimal solution for Problem I

Let us consider the topology in Fig. 3(a). The victim is connected to the FR a . A couple of users are connected to the FR h and FR f . The victim v observes that when traffic from u_1 follows the paths $h \leftrightarrow g \leftrightarrow d \leftrightarrow a$ or $h \leftrightarrow e \leftrightarrow d \leftrightarrow a$, the data rate falls below a threshold. Similarly, if the traffic from u_2 follows $f \leftrightarrow e \leftrightarrow d \leftrightarrow a$ the data rate falls below the threshold. On the other hand, the rate is good if the traffic from u_1 and u_2 follow $h \leftrightarrow e \leftrightarrow b \leftrightarrow a$ and $f \leftrightarrow c \leftrightarrow b \leftrightarrow a$, respectively. From this observation, v finds out that links 3, 6, 7, 8, and 11 are PCLs. Because of limited knowledge or access, the v cannot detect the actual congested links. A victim in an autonomous system might not know the status of a link in other autonomous systems. For example, this situation could happen if only links 3, 6, and 8, or 7 and 11 are congested. The v does not want to compromise data rates from its users and wants all the legitimate traffic coming to it to follow non-congested paths. We color the congested links red and non-congested links green. The black links do not carry any legitimate traffic and are assumed to be non-congested. We divide the process of determining the minimum number of filter assignments into four steps:

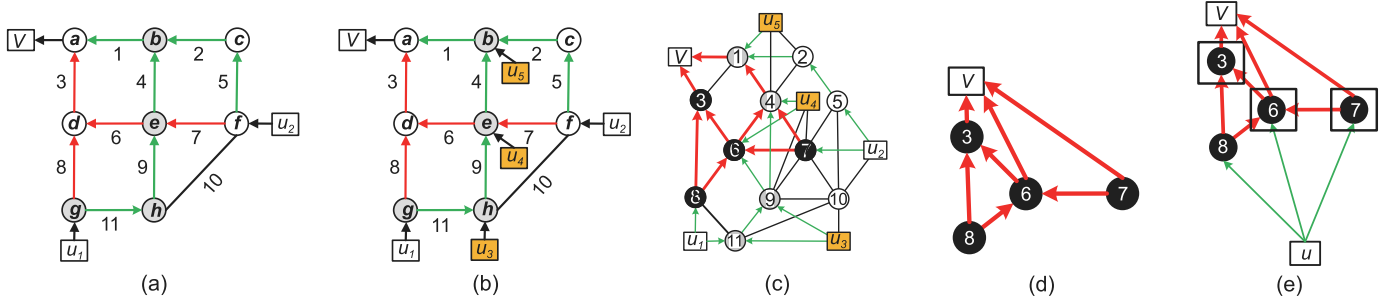


Fig. 4. Problem II example. (a) Original topology G . (b) Virtual user added in G . (c) G' with flow. (d) G' with black nodes and connecting links. (e) Final graph G'' .

Algorithm 2 Create flows in the line graph G'

Input: Topology G' .

```

1: Procedure: CREATE-FLOW( $G'$ )
2:    $u \leftarrow$  all users in  $G'$ 
3:   while  $u \neq \emptyset$  do
4:     for all  $n \in u$  do
5:        $nh \leftarrow \{n' : n' \text{ offers shortest paths to } n\}$ 
6:       if  $C_n(n) = \text{black}$  or  $C_f(n) = \text{red}$  then
7:          $\forall_{n' \in nh} C_e(e_{nn'}) \leftarrow \text{red}$ 
8:          $\forall_{n' \in nh} C_f(n') \leftarrow \text{red}$ 
9:       else if  $C_e(e_{nn'}) \neq \text{red}$  then
10:         $\forall_{n' \in nh} C_e(e_{nn'}) \leftarrow \text{green}$ 
11:         $\forall_{n' \in nh} C_f(n') \leftarrow \text{green}$ 
12:      $u' \leftarrow u' \cup nh$ 
13:    $u \leftarrow u'$ 

```

- Step 1: Transformation: In this step, we transform the topology graph G to line graph G' .
- Step 2: Edge coloring: In this step, we create flows from users that follow the shortest paths and we color the edges green or red.
- Step 3: Elimination: In this step, create graph G'' by eliminating white nodes.
- Step 4: Min separation: We apply the graph separation algorithm to find a min-cut in G'' .

In Step 1, we transform the original graph $G = (V, E)$ to line graph $G' = (V', E')$. V is the set of FRs and E is the set of links in the topology. The edges in G become the nodes in G' . Two nodes are neighbors in G' if the corresponding links in G are adjacent. We color the corresponding node of a PCL as black and other nodes as white. In Fig. 3(a), link 6 is connected to d and e so that all the links which are connected to d {3, 8} and e {4, 7, 9} will be neighbor of 6 in G' . Therefore, node 6 is connected to 3, 8, 4, 7, and 9 in G' . In G , v is connected to a and a is connected with two other links {1, 3}, therefore v will be connected to node 3 and 1 in G' . Similarly, we add users to G' . The color of nodes 3, 6, 7, 8, and 11 is black in G' because the links 3, 6, 7, 8, and 11 are possible congested G. Fig. 3(b) shows the line graph G' . In Step 2, we create traffic flow from every user in G' . The flow travels through all the next hops remaining on the shortest paths to the victim. The flow continues and the traveled links are colored green. When a flow encounters a black node, the flow becomes red and continues coloring the traveled links red. If a link is already colored red, the color never changes, but if it is marked as green, then a red flow changes the link color to red. The flow terminates at the victim. The process is shown in Alg. 2. In Fig. 3(c) the green flows travel nodes 11 and 9 which remain on the shortest paths from u_1 to v . The edges $u_1 \leftrightarrow 11$ and $u_1 \leftrightarrow 9$

become green. Since 11 is a black node, the flow becomes red and the edge $11 \leftrightarrow 8$ becomes red. The process continues until all the flows reach v . In Step 3, we create another graph $G'' = (V'', E'')$ with the victim node, red links, and nodes which are endpoints of the red links. Then, we eliminate the white nodes from G'' . When we eliminate a white node we connect its incoming neighbors (who are sending flow to the white node) with the outgoing neighbors (who are receiving flows from the white node). Alg. 3 shows this step in detail. For example, when we eliminate node 1 we connect v and 4 by $4 \leftrightarrow v$. Similarly, we connect v and 7 by $7 \leftrightarrow v$ when we remove node 4. We create a virtual user u and edges from u to any black node that is connected to at least one green edge. Any black node that appears first on the shortest path from any user to the victim has a green incident edge. According to Fig. 3(c), nodes 11, 6, and 7 will be connected to the virtual source. Now G'' contains only source, victim, and black nodes. Graph G'' represents all the congested shortest path flow from every user to the victim. Fig. 3(e) shows G'' . In Step 4, a minimum number of nodes are selected to separate the victim and virtual user in G'' using Acid and Campos's minimum d-separating set [22]. When we are separating the victim and the virtual source, we are cutting all the shortest possible congested paths with a minimum number of link blockages. Applying Acid and Campos's method we find the minimum separation set {3, 7}. Therefore, if we block links 3 and 7 in G by sending filters to FR d and f , no legitimate traffic will follow any congested path. The complete procedure is shown in Alg. 1.

Theorem 1. Alg. 1 finds optimal number of links to block.

Proof. Alg. 1 transforms the original topology to a flow graph where all shortest congested flows are considered. According to the assumption, there is at least a non-congested shortest path. The flows travel one of the non-congested shortest paths after disabling all the possible congested paths. To prove the optimality of the Alg. 1, we need to prove that G'' contains all the possible congested flows. Assume there is another red path from a black node n to v which does not use any link in the E'' . Then, there might be another node n' which was not selected for forwarding flow during flow creation process in Step 5 of Alg. 2. This means n' offers a path longer than the shortest path. FR n will never select a next-hop offering a longer path when it already has a neighbor offering the shortest path. Therefore, there are no other congested flows from a user which is not in G'' . Cutting all the flows by removing some nodes means blocking the congested shortest paths in G . As Acid and Campos's method provides the minimum size vertex cut set, the algorithm will return the optimal solution.

Theorem 2. The complexity of Alg. 1 is $O(m^3d)$.

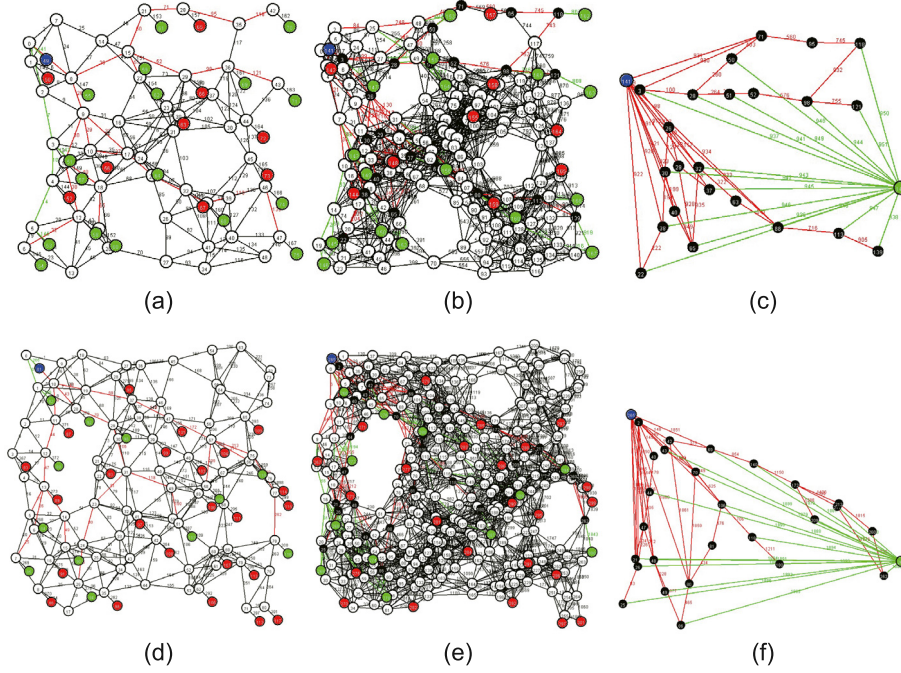


Fig. 5. Topologies I and II. (Green node=user, red node=bot, blue node=victim, red link=PCL). (a) Topology I. (b) G' for Topology I. (c) G'' for Topology I. (d) Topology II. (e) G' for Topology II. (f) G'' for Topology II.

Proof. Let us assume that G has n nodes, m edges, and d maximum node degree. Step 1 stage takes $O(md)$ time. If d is the maximum node degree in G , then G' has $2d - 1$ max node degree which is $O(d)$. Step 2 takes $O(md)$ to create flow for a user. In the worst case the number of users can be m and Step 2 takes $O(m^2d)$ time. To create G'' from G' , $O(md)$ time is needed in the worst case. Therefore, the conversion from G to G'' takes $O(m^2d)$. Step 4 takes $O(m^3d)$ to find a minimum vertex separator. Therefore, the algorithm takes $O(m^3d)$.

Algorithm 3 Elimination of white nodes from the flow graph G'

Input: Topology G' .

Output: The final graph G'' .

```

1: Procedure: ELIMINATE( $G'$ )
2:    $G'' \leftarrow G'$ 
3:   for all  $n \in V'$  do
4:     if no incident edge to  $n$  is red then
5:       Remove  $n$  from  $V''$ 
6:        $\forall_{j \in \text{Neighbor}(n)}$  remove  $e_{nj}$  from  $E''$ 
7:     else if  $C_n(n) = \text{white}$  then
8:       Remove  $n$  from  $V''$ 
9:       Add edges from incoming to outgoing neighbors of
10:       $n$ 
11:   Create a virtual node  $u$ 
12:   for all nodes  $n$  in  $V''$  do
13:     if # of green incident edge to  $n \geq 1$  then
14:       Add edge  $e_{un}$  to  $E''$ 

```

6. Extension of Problem I

In this section, we formulate a similar problem for a more general scenario and provide an optimal solution. In Problem I, we consider that there is at least the shortest non-congested path from every user. In some cases, all of the shortest paths and some paths longer than the shortest path can become congested. Alg. 1 fails to solve the problem in this situation. Therefore, we formulate Problem II, in which we assume that there exists a non-congested path from each user to victim.

Problem II. Find a minimum number of filter assignments so that all legitimate traffic follows non-congested paths.

In this problem, we assume that from every user there is a non-congested path. Let P_u be the set of paths from a user u to the victim v . After deploying B filters to FRs, we get $P_u^s(B)$ where $P_u^s(B) \not\subseteq P_u$ and none of the paths in $P_u^s(B)$ use any PCLs.

Therefore, the problem can be expressed as the following optimization problem:

$$\begin{aligned} & \text{minimize } B \\ & \text{subject to } \forall_{1 \leq u \leq U, p \in P_u^s(B)} p \cap L_c = \emptyset \end{aligned} \quad (2)$$

7. An optimal solution for Problem II

Let us consider the topology in Fig. 4(a). From the FR g , there is only one shortest path that contains PCLs. Therefore, the traffic from u_1 should go through the shortest detour to avoid the congested links. We divide the procedure to find the minimum number of links to block into five steps:

- Step 1: Virtual user creation: In this step, we find the nodes that are on a detour and add virtual users.
- Step 2: Transformation: same as Step 1 in Problem I
- Step 3: Modified edge coloring: This step is a simple modification of step 2 in Problem II.
- Step 4: Elimination: same as Step 3 in Problem I.
- Step 5: Min separation: same as Step 4 in Problem I.

In Step 1, we compute the shortest path distance and next hops from each user whose shortest paths are all congested, ignoring the PCLs. We color the nodes that are on the shortest paths gray. In Fig. 4(a), ignoring the PCLs (3, 6, 7, and 8), we get the detour from g which is $g \leftrightarrow h \leftrightarrow e \leftrightarrow b \leftrightarrow a$. We color $g, h, e,$ and b as gray and ignore the node a which is connected to the victim. This is because, when a packet reaches a it will always reach the victim without any trouble. Then, we add a virtual user to each gray node. $u_3, u_4,$ and u_5 are virtual users. It is clear that when a packet arrives at a gray node, there might be other possible congested paths to the victim that have equal

or less length compared to the detour. The gray FR might choose a shorter path than the detour. If there is no shorter path than the detour, then there is a possibility to choose one of the congested paths by an FR. Therefore, we want to block all the congested shortest paths from every gray node by using a modified version of the previously described method. In Step 2, we transform G to G' using Step 1 of [Problem 1](#). In Step 3, green flows are created from every user (both real and imaginary). Unlike Step 2 of [Problem 1](#), here, the flow travels through the paths that are less than or equal to the length of the detour. For example, in [Fig. 4\(b\)](#), from the gray node e the detour $e \leftrightarrow b \leftrightarrow a$ is 3 hops. The neighbors that offer the shortest path less than or equal to 2 are selected, and flow propagates to those nodes. The nodes d and b offer 2 length paths. The flow travels to d and b . When the flow reaches b , the traveled hop is 1 and we look for neighbors of b that offer the shortest path less than or equal to 1. The flow continues this way. This way the flow travels in all possible ways which could destruct the user traffic passing through the detour. Due to space limitation, the procedure $\text{CREATE-FLOW-2}(G')$ is not shown here. The green flows from each virtual user ensure a green incident edge to each node they travel first. This ensures the presence of green flow in each detour node. Therefore, it is important to add virtual users at each detour node. Otherwise, red flows from other users may overwrite the color of the green incident edge to a black/gray node. Step 4 and Step 5 are the same as Step 3 and Step 4 in [Problem 1](#). According to [Fig. 4\(e\)](#), the minimum vertex separation in G' is $\{3, 6, 7\}$. Blocking the edges 3, 6, and 7 by sending filters to the FRs d , e , and f ensures the shortest non-congested paths from each legitimate to the victim. The complete procedure is shown in [Alg. 4](#).

Theorem 3. The complexity of [Alg. 4](#) is $O(m^3 + n)$.

Proof. [Alg. 4](#) has an additional computation for adding imaginary users. The procedure takes $O(nd)$ to find out the detour nodes for all users. Then, adding virtual users takes $O(n)$ time. The modified flow generation procedure takes a similar amount of time.

Therefore, [Alg. 4](#) takes $O(m^3d + nd)$. If we consider mesh topology or other networks where the maximum node degree is small, then [Alg. 4](#) takes $O(m^3 + n)$.

Theorem 4. [Alg. 4](#) finds the optimal number of links to block.

Proof. [Alg. 4](#) transforms the original topology to a flow graph where all congested flows are considered. Disconnecting the source from the victim will disable all possible congested paths which are not longer than the detour. To prove the optimality of [Alg. 4](#), we need to prove that G'' represents all possible congested flow not longer than the detour.

Assume that there is another red path from a black node n to v which does not use any links in E'' . Then, there might be another node n' which was not selected for forwarding flow during the flow creation process in Step 5 of [Alg. 2](#). This means n' offers the shortest path longer than the detour. The FR n will never select a next-hop offering a longer path when it already has a neighbor offering a shorter path. Therefore, there are no other congested flows from a user that are not in G'' . Cutting all the flows by removing some nodes means blocking the congested paths in G . Acid and Campos's method provides the minimum size vertex cut set. Therefore, the algorithm will return the optimal solution.

8. Simulations

In this section, we present our experimental setting and simulation results comparing some existing works.

8.1. Simulation settings

We conducted all the experiments with a custom-built Java simulator. The main reason for using a custom-built simulator is its scalability. We do not need to analyze transmission time or natural packet drop issues. We only need to count the number of legitimate received packets, number of PCLs, number of links to block, and number of hops to the victim. The network topologies we considered contain 68 – 2000 routers. Using NS3 or other similar simulators for this kind of simulation would take several days. That is why we built our own Java multi-threaded simulator to get the results quickly.

We conducted simulations for randomly generated graph topologies (Topology I and Topology II) and a subset of real network topology (Topology III). We used two randomly generated topologies with node degrees between 1 and 15 and internal node user probabilities between 0.2 and 0.5. To generate the topology we select an area of 500×500 square units. Then we divide the area into 50×50 blocks. In each block, a certain number of nodes are placed at random locations. We also ensure that two nodes are at least a certain distance away. The edges are generated based on the distances among nodes. If a node is closer than a certain distance to another node, we connect them with a link. Finally, we add a certain number of users and bots to some nodes. The number of users and bots were selected randomly from a uniform distribution. The links passing traffic of a large number of users are selected as the target links. In some cases, the target link selection is random.

If a topology has an internal user probability of 0.5, it means that 50% of the nodes are attached to attackers or users. The Topology III was taken from the Autonomous systems AS-733 dataset [23]. The details are shown in [Table 2](#). Topologies I and II are depicted in [Figs. 5\(a\)](#) and [5\(d\)](#). The green, red, blue, and white nodes represent the users, bots, victims, and FRs, respectively. The red links represent the PCLs. The line graph G' with flows for Topologies I and II are shown in [Figs. 5\(b\)](#) and [5\(e\)](#). The black nodes represent the congested links. As there is no effect of bots in coloring the edges, we ignore the red nodes when creating flows. The final flow graphs G'' are shown in [Figs. 5\(c\)](#) and [5\(f\)](#). The black rectangles around a black node represent that the corresponding links of that node are selected for blocking.

Algorithm 4 Find a minimum number of blockage

Input: Topology G , PCLs L_c .

Output: A set of links to block from L_c .

```

1: Procedure: FIND-ASSIGNMENT-2( $G$ )
2:   ADD-IMAGINARY- $U(G)$ 
3:    $G' \leftarrow$  the line graph of  $G$ .
4:    $\text{CREATE-FLOW-2}(G')$ 
5:    $G'' \leftarrow \text{ELIMINATE}(G')$ 
6:    $g \leftarrow$  minimum vertex separation set in  $G''$ 
7:   return  $g$ 
8: Procedure: ADD-IMAGINARY- $U(G)$ 
9:   Remove all links in  $L_c$  from  $E$ 
10:  for  $n \in V$  do
11:    if all shortest path of  $n$  are congested then
12:       $Q \leftarrow \{n\}$  ▷ initialize a queue with  $n$ .
13:      while  $Q \neq \emptyset$  do
14:         $n' \leftarrow \text{DEQUEUE}(Q)$ .
15:         $C_n(n') \leftarrow \text{gray}$ .
16:         $n'' \leftarrow$  neighbors of  $n'$  offering shortest path.
17:         $\text{ENQUEUE}(Q, n'')$ .
18:  Add all links in  $L_c$  to  $E$ 

```

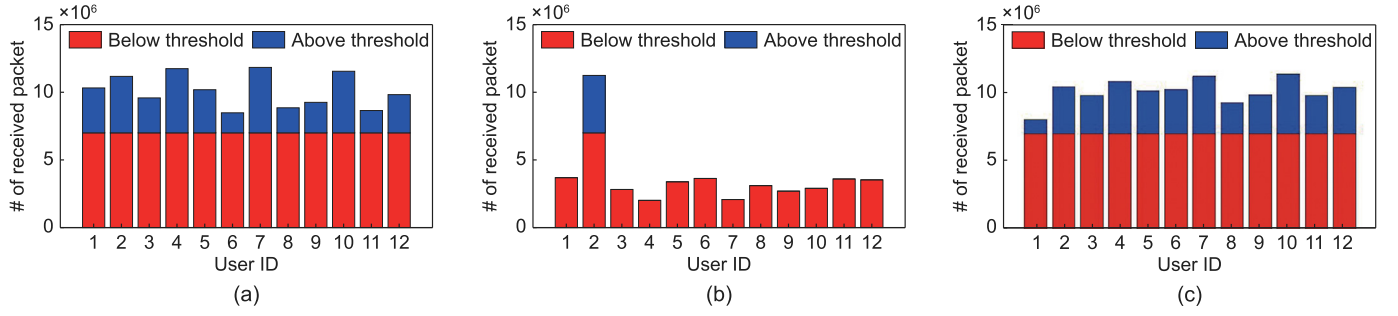


Fig. 6. Packet distribution of different scenarios in Topology II. (a) Before attack number of received packets. (b) During attack number of received packets. (c) During defense number of received packets.

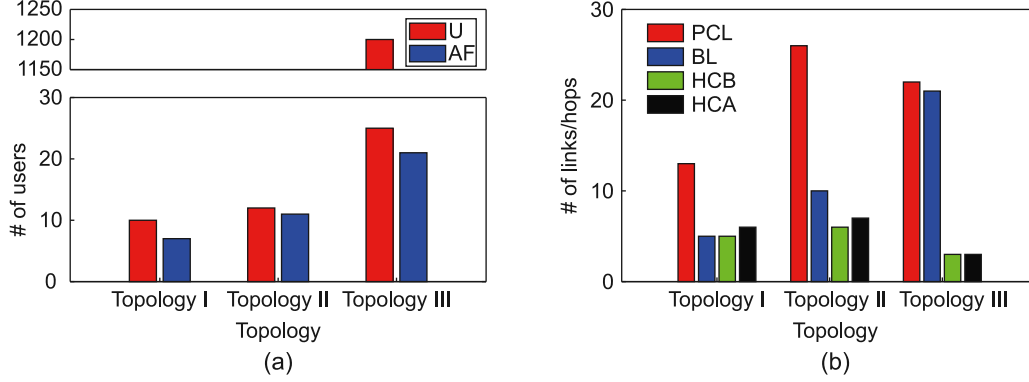


Fig. 7. Simulation results. (a) Total users and affected users. (b) Comparing properties of three topologies.

Table 2

Topology parameters.

Parameters	Topology I	Topology II	Topology III
Number of nodes	68	117	2000
Number of edges	179	302	10 800
Internal user probability	0.2	0.5	0.5
Max node degree	10	15	1459

8.2. Simulation results

As the bots are distributed uniformly over the network, the attacker can congest any of the links in the network. We assume that the attacker is capable of congesting only one link. It selects the link which carries traffic from the highest number of users. Simulations shown in Fig. 6 are conducted with this setting.

Fig. 6(a) shows the regular number of received packets by the victim from the users in Topology II. There are 12 users and the number of received packets from all of them is above the threshold (70,000,000) for a particular period. When the attack is started, the numbers of received packets from some of the users are reduced. From Fig. 6(b), we can see that the numbers of received packets of 11 users fall below the threshold. We also see in Fig. 7(a) that 11 users are affected by the attack. When the filters are deployed, we can see that the number of received packets by all of the users is above the threshold (see Fig. 6(c)).

Fig. 7(a) shows the number of affected users (AF) and total users (U). In Topology I, we see that after congestion, 7 of the 10 users become affected. For Topology II, 11 out of 12 users are affected. The ratio of the affected users in Topology III is less than the other topologies. This is because Topology III has a higher node degree and many shortest paths. Therefore, the traffic from different users goes through many distinct paths.

Fig. 7(b) shows PCLs, the number of links needed to block to avoid all PCLs (BL), and the average number of hops to reach the victim before filter deployment (HCB) and after filter deployment (HCA). We can observe that the BL is much smaller than the PCL. Especially in Topology III, the number of PCLs is 40, but they can all be avoided by blocking only 2 links. After deploying the filters, the average number of hops to the victim increases in Topology I but remains the same for Topologies II and III.

Figs. 8(a) and 8(b) show the number of affected users, PCLs, blocked links, and hop counts after filter deployment. We observe that all the measurements increase with a higher number of attacked links. The HCA does not increase in Topology III, but it increases by 1 or 2 in Topology I. This is because Topology III has more redundant connections than Topology I.

Fig. 8(c) shows the comparison between our proposed FR-based model and the BGP advertisement-based model. It is hard to compare these two models because the BGP advertisement-based model assumes that the deployer autonomous system knows exactly which links are congested. We assume that the victim (deployer) does not know exactly which links are congested, which is more practical. We assume that the victim in the FR-based model knows the congested links to give both models equal information. The BGP advertisement-based model will block all of the congested links (it avoids the congested links by false path advertisement) but our FR-based model blocks much fewer links than the BGP advertisement-based model. Fig. 8(c) shows the number of blocked links by the number of attacked links for both approaches. The attacked links are chosen randomly and an average of several rounds of simulation are shown in the figures. When the number of congested links is 70, the FR-based model can reduce 43% congestion compared to the SPIFFY approach. When the number of congested links is above 250, the FR-based

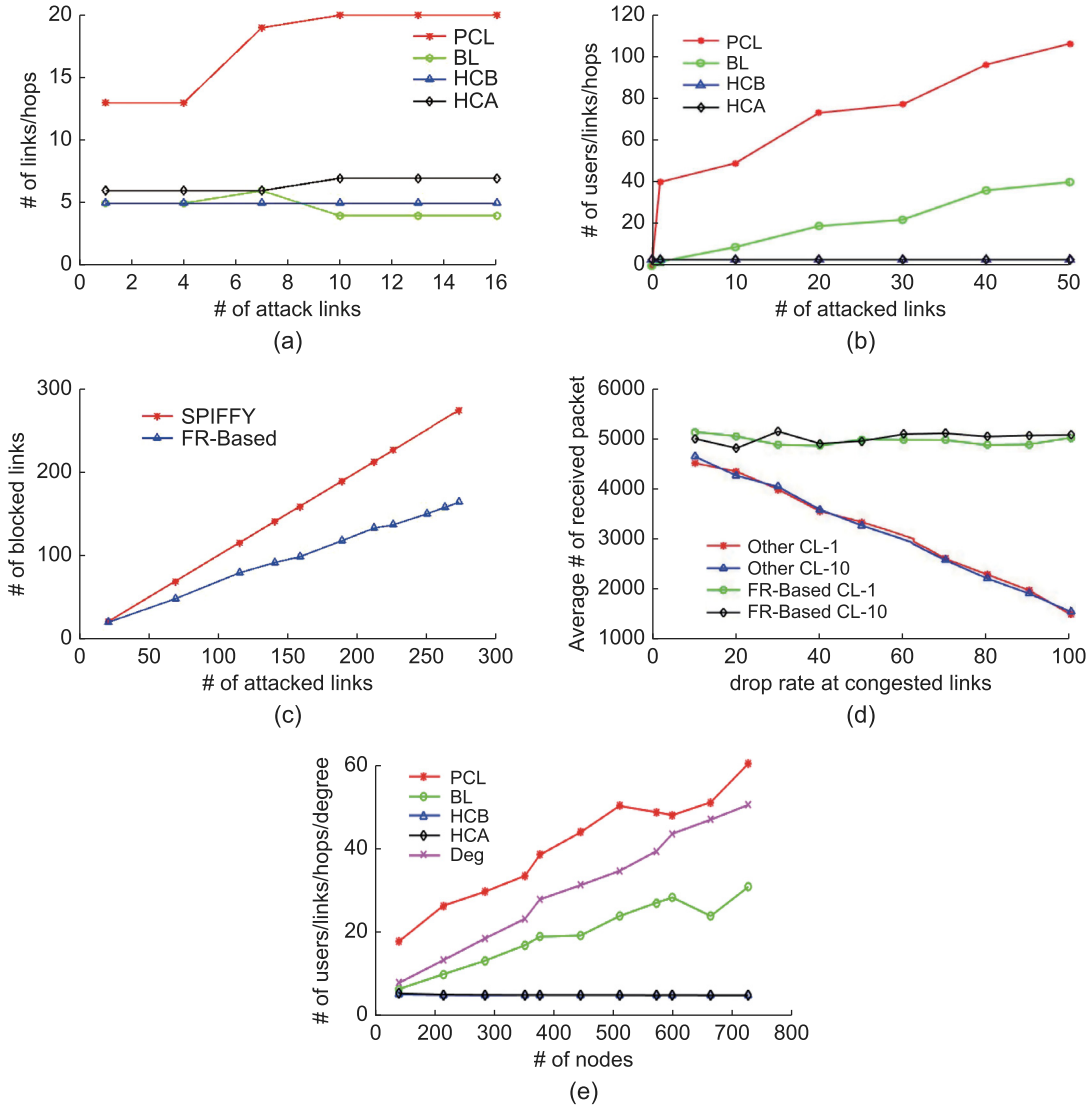


Fig. 8. More simulation results. (a) Different measurements for Topology I. (b) Different measurements for Topology III. (c) Comparison with BGP advertisement based method for Topology II. (d) Comparison with router detection-based methods for Topology III. (e) Different number of nodes.

model can reduce above 66% congestion compared to the SPIFFY approach.

We compare the FR-based model with some other approaches including Floc [16], SPIFFY [13], ColDef [14], which use attack traffic detection at the router. We show the detrimental effects when the routers cannot detect the attack traffic. Fig. 8(d) shows the average number of received packets by the ratio of packet drop at congested links. The average number of received packets is reduced by the drop rate at congested links. The number of received packets for 10 congested links is slightly less than that of one congested link. The numbers of received packets are similar regardless of the number of congested links or drop rate for the FR-based model. This is because when filters are deployed, none of the user traffic goes through any congested links.

Fig. 8(e) shows PCLs, BLs, HCB, degree, and HCA of randomly generated topologies with different node degrees. The average node degree increases with the number of nodes. This is because the node density increases with the number of nodes and the neighborhood radius remains the same. We also observe that the BL is much smaller than the PCL. The difference increases with the increase of number of nodes. This is because when

the topology is large there are more ways to congest links that yield the same packet rates of the affected flows. The number of PCLs and BLs increases with the topology size. The HCA and HCB remain the same because of many alternative shortest paths in large topologies.

From the above experimental results, we can conclude that the FR-based model can efficiently protect the users' traffic. We need to block a few links to avoid a large number of PCLs. The model works with limited knowledge about the status of the network and can protect its users at the minimum cost.

9. Discussion and future work

Nowadays, internet service providers (ISPs) usually do not use the shortest path routing. The routing is influenced by congestion and relations with other ISPs. The routing and network engineering are designed to optimize the congestion based on the demand of their customers. Our proposed FR-based model can easily adopt any routing protocol. Changing the route of a destination may cause some overhead to the ISP. They should be aware of this issue and should not accept filters that lower

their QoS. Further research is needed to identify whether or not a filter is harmful. The approach is scalable to attacks of any size. The problems consider multiple congested links and the scenario where there are no alternative shortest paths. Therefore, a large attack will make several paths congested and may block all the shortest paths. On the other hand, the complexity of the algorithm depends on the topology size but not on the number of PCLs. Therefore, because different attacks have different sizes, the complexity/run time does not change.

In this work, we consider a victim in a network. In reality, there can be multiple victims. According to our assumptions, all the victims protect their users by deploying filters separately. There is no victim to victim communication. This method is not efficient when multiple victims want to protect a group of users. Then a filter may affect traffic to multiple victims. If the filter's condition is a prefix of address instead of an address, then the victims that have the prefix in their address will be able to direct traffic that is coming to any of the victims with that prefix in address. This system requires that the victims having the prefix will cooperate and may be owned by an entity. Existing internet service providers or autonomous systems have similar IP allocations. In that case, we need to generate a combined victim like the combined user in Step 3. The rest of the procedures will be the same. Therefore, in the future, we may extend the problem as described in this paper to solve the multiple victim scenarios.

10. Conclusion

The link flooding attack is the most powerful attack that makes a service unavailable to legitimate users. It is not possible to protect any server from DDoS attacks without the help of the network equipment. The current DDoS defense system cannot protect the link flooding attack because the attack packets do not go to the victim. Therefore, the victim remains unaware of the location of the attacker. Routers which are the most important component in the network can be upgraded to FRs easily. Besides, the FR can work in a network with legacy routers. An ISP can gain benefits economically by deploying FRs. Therefore, both the deployer and victim get benefited from this model. In this work, we present an optimal filter assignment policy. We compare the performances of the proposed assignment policy with some existing approaches. Simulation results show that our proposed approaches work much better than the other existing approaches. In the future, we may formulate another problem to find an optimal assignment for multiple cooperative victims.

CRedit authorship contribution statement

Rajorshi Biswas: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Jie Wu:** Conceptualization, Methodology, Formal analysis, Investigation, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Wei Chang:** Writing – review & editing. **Pouya Ostovari:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research was supported in part by the NSF grants (CNS 1757533, CNS 1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS1460971, and IIP 1439672).

References

- [1] J. Caballero, C. Grier, C. Kreibich, V. Paxson, Measuring pay-per-install: The commoditization of malware distribution, in: Proceedings of the 20th USENIX Conference on Security, USENIX Association, Berkeley, CA, USA, 2011.
- [2] M.S. Kang, V.D. Gligor, V. Sekar, Defending against evolving DDoS attacks: A case study using link flooding incidents, in: J. Anderson, V. Matyáš, B. Christianson, F. Stajano (Eds.), Security Protocols XXIV, Springer International Publishing, 2017.
- [3] D. Seo, H. Lee, A. Perrig, PFS: Probabilistic filter scheduling against distributed denial-of-service attacks, in: 2011 IEEE 36th Conference on Local Computer Networks, 2011.
- [4] D. Seo, H. Lee, A. Perrig, APFS: Adaptive Probabilistic Filter Scheduling against distributed denial-of-service attacks, *Comput. Secur.* 39 (2013).
- [5] R. Biswas, J. Wu, Filter assignment policy against distributed denial-of-service attack, in: 2018 IEEE 24th International Conference on Parallel and Distributed Systems, ICPADS, IEEE, 2018.
- [6] J. Wang, I.C. Paschalidis, Statistical traffic anomaly detection in time-varying communication networks, *IEEE Trans. Control Netw. Syst.* 2 (2) (2015).
- [7] A. Dahiya, B.B. Gupta, A reputation score policy and Bayesian game theory based incentivized mechanism for DDoS attacks mitigation and cyber defense, *Future Gener. Comput. Syst.* 117 (2021) 193–204.
- [8] A. Dahiya, B. Gupta, Multi attribute auction based incentivized solution against DDoS attacks, *Comput. Secur.* 92 (2020) 101763.
- [9] K. Argyraki, D.R. Cheriton, Loose source routing as a mechanism for traffic policies, in: Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture, ACM, New York, NY, USA, 2004.
- [10] M. Motiwala, M. Elmore, N. Feamster, S. Vempala, Path splicing, in: SIGCOMM Comput. Commun. Rev., ACM, New York, NY, USA, 2008.
- [11] W. Xu, J. Rexford, MIRO: Multi-path interdomain routing, in: Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2006.
- [12] X. Yang, D. Wetherall, Source selectable path diversity via routing deflections, *SIGCOMM Comput. Commun. Rev.* 36 (4) (2006).
- [13] M. Suk Kang, V. D. Gligor, V. Sekar, SPIFFY: Inducing cost-detectability tradeoffs for persistent link-flooding attacks, in: Network and Distributed System Security Symposium, 2016.
- [14] S.B. Lee, M.S. Kang, V.D. Gligor, CoDef: Collaborative defense against large-scale link-flooding attacks, in: Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies, ACM, New York, NY, USA, 2013.
- [15] J. Ioannidis, S. M. Bellovin, Implementing pushback: Router-based defense against DDoS attacks, in: Network and Distributed System Security Symposium, 2002.
- [16] S.B. Lee, V.D. Gligor, Floc : Dependable link access for legitimate traffic in flooding attacks, in: 2010 IEEE 30th International Conference on Distributed Computing Systems, 2010.
- [17] A. Dutta, E. Al-Shaer, S. Chatterjee, Q. Duan, Autonomous cyber defense against dynamic multi-strategy infrastructural DDoS attacks, in: 2023 IEEE Conference on Communications and Network Security, CNS, 2023, pp. 1–9.
- [18] S. Neelavathy Pari, E.C. Ritika, B. Ragul, M. Bharath, AI-based network flooding attack detection in SDN using multiple learning models and controller, in: 2023 12th International Conference on Advanced Computing, ICoAC, 2023, pp. 1–7.
- [19] J.M. Smith, M. Schuchard, Routing around congestion: Defeating DDoS attacks and adverse network conditions via reactive BGP routing, in: IEEE Symposium on Security and Privacy, 2018.
- [20] R. Biswas, J. Wu, Optimal filter assignment policy against distributed denial-of-service attack, *IEEE Trans. Dependable Secure Comput.* (2020) 1, <http://dx.doi.org/10.1109/TDSC.2020.2987301>.
- [21] R. Biswas, J. Wu, W. Chang, P. Ostovari, Optimal filter assignment policy against transit-link distributed denial-of-service attack, in: 2019 IEEE Global Communications Conference, GLOBECOM, 2019, pp. 1–6.
- [22] S. Acid, L.M. De Campos, An algorithm for finding minimum D-separating sets in belief networks, in: Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 1996.
- [23] Stanford, Autonomous systems AS-733, <https://snap.stanford.edu/data/as-733.html>.