2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)

Filter Assignment Policy Against Distributed Denial-of-Service Attack

Rajorshi Biswas and Jie Wu
Department of Computer and Information Sciences
Temple University, Philadelphia, PA, USA
{rajorshi, jiewu}@temple.edu

***Abstract*—A denial-of-service (DoS) attack is a cyber-attack in which the attacker sends out a huge number of requests to exhaust the capacity of a server, so that it can no longer serve incoming requests and DoS occurs. The most devastating distributed DoS attack is performed by malicious programs called bots. With the help of a special type of router called filter router, the victim can protect itself and reduce useless congestion in the network. A server can send out filters to filter routers for blocking attack traffic. The victim needs to select a subset of filter routers wisely to minimize attack traffic and blockage of legitimate users (LUs). In this paper, we formulate two problems for selecting filter routers given a constraint on the number of filters. The first problem considers the source-based filter and we provide greedy approximation solutions. The second problem considers the destination-based filter and how to minimize total amount of attack traffic and blocked LUs. We propose a dynamic programming solution for the second problem. We present simulation results comparing the proposed solutions with a naive approach. Our simulation results strengthen support for our solutions.**

***Index Terms*—botnet, DDoS defense, DDoS, flooding attack, filter router, network security**

I. Introduction

A denial-of-service attack (DoS attack) is a cyber-attack in which the attacker seeks to make a machine or network resource unavailable temporarily to its users. DoS attacks are considered a federal crime under the Computer Fraud and Abuse Act with penalties that include years of imprisonment [1]. The Computer Crime and Intellectual Property Section of the US Department of Justice handles cases of DoS attacks. Therefore, detecting DoS attacks and identifying attackers have been an important issue in Network Forensics. Moreover, DoS attacks are increasing day by day in both number and size; CloudFlare [2] recently reported a 400 Gbps massive DoS attack that took place at their servers.

There are several types of DoS attacks including SYN Floods, Malformed Packets, UDP Floods, Amplification Attacks, and Distributed Attacks [3]. The objective of DDoS is to generate a lot of packets from different locations to exhaust incoming/outgoing bandwidth of the victim. A coordinator would send commands to workers who continue to send requests to the target. The workers are known as bots and the network of workers is known as botnet. As normal users also request through the NAT, it is difficult for the victim to differentiate between the bot requests and normal user requests. Fig. 1 shows the DDoS attack model by botnet. An effective method of preventing DDoS attack is to use filter routers (FRs) in network infrastructure. FRs are a special type of router which is capable of packet marking and receiving

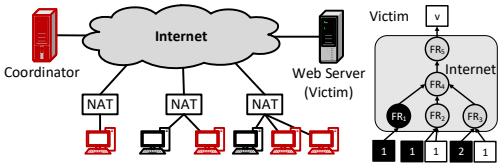


Fig. 1. DDoS attack by bots.

filter tasks. Packet marking task refers to attaching the FR's own IP address probabilistically to the packets it forwards while receiving filter task refers to receiving filters from a web server. A web server can block all or part of the traffic destined to it. The packet marking is used to find the topology by the victim. After topology construction, the victim generates filters and selects a subset of the FRs to assign them.

In this paper, we focus on finding optimal filter assignment assuming that the victim has already constructed the traffic topology. We formulate two problems and propose solutions for them. In the first problem, a limited number of source-based filters are assigned to the FRs. For example, if the victim can assign 2 filters, it can select $\{FR_1, FR_2\}$, $\{FR_1, FR_5\}$, $\{FR_2, FR_4\}$ or another pair of FRs (see Fig. 1). If the victim selects the first pair of FRs, the attack traffic from FR_3 will reach the victim which is highly unexpected. If the second pair is selected then the attack traffic will travel through (FR_2, FR_4), (FR_3, FR_4), and (FR_4, FR_5) links. The amount of attack traffic in each link is not same. It is challenging to find a filter assignment for which the total amount of attack traffic is minimum. We propose greedy approximation solutions for this problem. In the second problem, a limited number of destination-based filters are assigned to the FRs. Destination-based filter blocks every packet at FR that is destined to the victim. If the victim selects the third pair, then all the legitimate users (LUs) will be blocked and the attack traffic will travel through (FR_2, FR_4) and (FR_3, FR_4) links. It is also challenging to find a filter assignment so that the total attack traffic and number of blocked LU are minimum. We propose a dynamic programming solution for this problem. Our main contributions are the following:

1) We formulate two problems for finding filter assignment with a budget (limited number of filters) and provide greedy and dynamic programming solutions.
2) We present simulation results to support our model.

The remainder of this paper is arranged as follows: Section II presents some related works and their limitations. In Section III, we present the system model for preventing DDoS attack.

978-1-5386-7308-9/18/$31.00 ©2018 IEEE
DOI 10.1109/ICPADS.2018.00076

Section IV presents the formal definition of the first problem and our proposed greedy solutions. Section V presents the formal definition of the second problem and our proposed dynamic programming solution. In Section VI, we present some simulation results that strengthen our proposed solutions.

II. Related Works

There exist many statistical methods including correlation, entropy, covariance, divergence, cross-correlation, and information gain to detect anomalous DDoS requests [4]. Authors in [5] introduced a model of randomized DDoS attacks with increasing emulation dictionary where the attackers use the attack definition from the dictionary that contains request patterns similar to those of LUs. They proposed an inference algorithm for identifying the botnets executing such DDoS attacks. Nowadays, static path identifiers are used for inter-domain routing objects, which makes it easy for attackers to launch the DDoS flooding attacks. In [6], the authors present a design dynamic path identification framework that uses path identifier negotiated between the neighboring domains as inter-domain routing objects.

In [7], the authors propose a method, RADAR, to detect and throttle DDoS attacks using adaptive correlation analysis on SDN switches. The system can defend against a wide range of flooding-based DDoS attacks including link flooding, SYN flooding, and UDP-based amplification attacks. In [8], the authors propose a new approach which reduces the resource utilization factor to a minimal value for quick absorption of the attack. In [9], a DDoS protection mechanism, SkyShield, is proposed by taking advantages of the sketch techniques. To identify malicious hosts efficiently, they used the abnormal sketch obtained from the last detection cycle. The SkyShield could leverage other techniques including Bloom filters and the CAPTCHA. In [10], the authors propose a collaborative DDoS mitigation network system in which a domain helps another domain. A domain can direct excessive traffic to other trusted external domains for DDoS filtering. The filtered clean traffic is forwarded back to the targeted domain.

Most of the existing works are mainly concerned about the service availability of the server. In fact, the attack traffic may cause huge network congestion and DoS. Therefore, these techniques cannot protect the network from being contaminated by attack traffic. A victim and network component collaboration based system can help in this case. A four-phase DDoS protection system is proposed in [11]. The victim generates filters and sends them to the upstream FRs. The FRs send the filters to its upstream FRs and thus the filters propagate to the effective FRs. An adaptive version of PFS is proposed in [12]. The system sends directly filters to the high capable FRs first, then the filters propagate to the effective FRs. However, these two systems cannot select the FRs optimally when there is a limitation on selecting FRs.

III. System Model

Our system is composed of legacy routers (LRs), network address translators (NATs), filter routers (FRs), attackers, legitimate users (LUs), and a victim (v). Fig. 2 shows the complete system model. In reality, there are multiple victims in a network but for simplicity of explanation we are considering a single victim. We assume that end users are connected to a FR or a LR through NAT. The FRs are a special kind of router which are capable of two functionalities. Firstly, it can do packet marking which is used to construct traffic topology at the victim. Secondly, it can receive filter from the victim and apply the filter to block the attack traffic according to the filter definition. There can be two types of filter: source-based and destination-based. The source-based filter specifies blocking of traffic based on source address. For example, a source-based filter can be understood: *if source address is X then discard the packet*. If we use a source-based filter at FR_3 (assume X and Y are the IP addresses of the NATs connected to FR_1 and FR_2, respectively) then FR_3 will discard packets coming from NAT-X but forward packets from the NAT-Y.

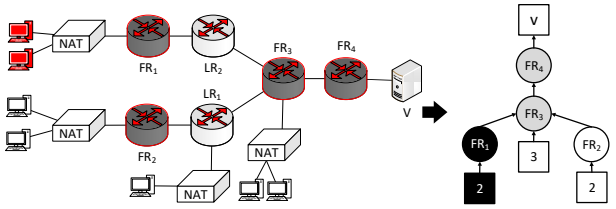


Fig. 2. System model and constructed topology.

The advantage of using source-based filter is that a FR can block the attack traffic by its source IP address and forward legitimate traffic. If the LU and attacker both remain behind the same NAT, then it is impossible to block only attack traffic. The limitation of the source-based filter is that it cannot protect if an attacker spoofs the IP address of a LU. If an attacker creates packet having Y as the source address, then the packet will not be blocked at FR_3. To protect against DDoS attack with IP spoofing, we can use destination-based filter. A destination-based filter is *if the destination address is X then discard the packet*. For example, if we use a destination-based filter at FR_3 (assume that X is IP address of v), then all the packets including legitimate and spoofed attack packets will be blocked by FR_3. The destination-based filter is more restrictive. When a FR uses it, it blocks all the attack and legitimate traffic destined for the victim. Therefore, spoofed attack traffic cannot penetrate.

The attackers are usually user devices which have compromised programs that can generate traffic destined for a target. The programs are controlled by a master. The master can send commands of attack to the program. This type of program is called bot and the network of bots is called botnet. Though the DDoS traffic is hard to differentiate from legitimate traffic, there exist several methods based on arrival time, packet size, and packet content for detecting attack packets [4]. In this paper, we are not focusing on the detection of attack packet and assume that the victim can find out the source address of attack traffic using these methods. The victim also knows the packet rate of each attacker. The complete protection process consists of the four phases.

In **phase 1**, the FRs probabilistically mark the packet it forwards by appending its own IP address. Assume the marking probability is 0.5. Then the victim v may get packets with $\{FR_1, FR_3\}$, $\{FR_3, FR_4\}$, or $\{FR_1, FR_4\}$. The victim may also get packets with $\{FR_2, FR_3\}$, $\{FR_3, FR_4\}$, or $\{FR_2, FR_4\}$. The $\{FR_1, FR_3\}$ marking indicates that the FR_1 remains before the FR_3 along the path from the user.

In **phase 2**, the victim constructs paths from all the sources after gathering enough information from marked packets.The victim can easily form a directly acyclic graph (DAG) combining all the paths. For simplicity we will consider tree instead of a DAG. We consider that bots and LUs are behind NAT of internet service provider. We color the bots/attacker as black and LUs as white. The FRs which forward the end users' traffic first are called *entry nodes*. $\{FR_1, FR_2\}$ are the entry nodes in Fig. 2. The FRs are colored as black, white, or gray. A black (or white) FR means it only forwards messages from attacker/bot (or LU). A gray FR forwards packets from both LU and attacker.

In **phase 3**, some of the FRs in the traffic topology are selected to assign filters. The traffic topology is simplified by removing nodes with no fork. A node having at least two children is called a fork node. Non-fork nodes are not efficient for assigning filters. Instead, selecting child node reduces attack traffic in the network. Therefore, an optimal filter assignment policy should select a set of FRs (g) from the set of gray and black nodes (G) with minimal blockage of legitimate traffic and contamination by attack traffic, while ensuring that no attack traffic can reach the victim. We define the contamination as the total attack traffic in the network. For example, if the attack traffic is blocked at FR_3 then total contamination is 2 (assume all attackers' packet rate is 1). We denote the contamination in a network for the g filter assignment set by W_c.

$$W_c(g) = \sum_{a=1}^{A} \alpha_a \times d_a,\ d_a = \min_{\forall_{n \in \text{PRED}(n) \cap g}} \text{dist}(a, n) \quad (1)$$

Here, PRED(n) is the set of predecessor of n, α_a is the traffic load of attacker a, dist(a, n) is the number of hops between a and n. A is the total number of attackers. Therefore, W_c is the total attack traffic load for selecting $|g|$ FRs out of $|G|$ FRs. If U is the set of LUs, then the number of blocked LUs for the filter assignment g is denoted by $U_b(g)$.

$$U_b(g) = \{u : u \in U \text{ and } \text{PRED}(u) \cap g \neq \emptyset\} \quad (2)$$

The best way to minimize blockage of LU and contamination is to block immediately after the attacker. In reality, there are a huge number of attackers and the victim needs to select a huge number of FRs to block them which is not possible. So, a victim has budget B of selecting a number of FRs. Therefore, $|g|$ should be less than or equal to B.

In **phase 4**, unused filters are removed from the FRs. As the FRs have limited capacity and computation power, it is necessary to reduce the workload by removing the filters. Besides, the attacker can flood the FRs by sending useless filters. This type of filters are evicted soon because they are most likely not being used.

The filter sent by a user (or victim) is only applicable to the packets which are destined for that user (or victim). However, an attacker can spoof IP of the victim and send wrong filters to FRs. This spoofed filter can be detected using a simple handshake protocol. The spoofing attacker will not be able to handshake with the spoofed IP address. However, we are not focusing on finding an optimal filter assignment policy which is discussed in the next section.

IV. Source-based Filter Assignment Policy

In this section, we formulate a problem of assigning source-based filters to the FRs so that the contamination is minimum.

A. ***Problem 1:*** *Find a filter assignment so that the contamination is minimal by ensuring that all the attack traffic is blocked before reaching the victim.*

In this problem, source-based filters are used. The contamination is defined by the total amount of attack traffic in the network. The problem can be expressed as the following:

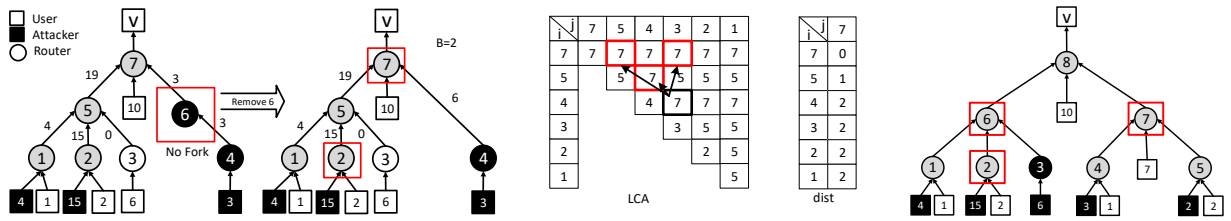
$$\begin{aligned} &\text{minimize} && W_c(g) \\ &\text{subject to} && |g| \leq B,\ \forall g \subset G,\ v \notin G \end{aligned} \quad (3)$$

The victim v will be white $(v \notin G)$ if all attacker traffic is blocked before reaching it. The complexity of optimal solution is unknown. Therefore, we propose and compare two greedy approximation solutions as follows.

B. First Greedy Solution

A filter is first assigned to root to guarantee the blockage of all attack traffic. The rest of the filters are assigned using a greedy approach. We first calculate weight of each node. The weight of a node is its attack flow times the distance from the node to the root, or the closest node having filter on the path to the root. Then the highest weighed node is selected for filter assignment and weight is updated accordingly. After every new assignment we need to check for each node in assignment set whether attack flows from all children are blocked by other filters or not. If all attack flows are blocked, then we remove the filter from the node. The process continues until the number of filters is less than the budget.

Let us consider the tree in Fig. 3(a) that is constructed from information from marked packet by the victim. Assume we want to find assignment of 3 filters ($B = 3$). We first assign a filter to 7. The weights of nodes 1, 2, 5, and 8 are $4 \times 2 = 8$, $15 \times 2 = 30$, $(15 + 4) \times 1 = 19$, and $3 \times 2 = 6$, respectively. So, node 2 is taken for assignment. Now new weight of 2 is 0 and new weight of 5 will be $4 \times 1 = 4$. None of $\{7, 2\}$ nodes' incoming attack traffic is blocked. Therefore, none of the nodes are removed from assignment. The next highest weight is of node 4. Therefore, the filter assignment is $\{7, 2, 4\}$. The W_c of this assignment is 8 while the optimal W_c is 0 for $B = 3$. For randomly generated trees, filter assignment produced by this greedy approximation is almost twice as much as than the optimal filter assignment. Details are shown in Section VI.



(a) Topology for Problem 1. (b) LCA calculation. (c) Topology for Problem 2.

Fig. 3. Topologies for problems 1 and 2.

Algorithm 1 Second Greedy Blocking Strategy

Input: The number of filters B, topology tree T
Output: A set of nodes in T

Procedure: BLOCK-GREEDY-2(B, T)
  Initialize LCA, $dist$, g and $B_c \leftarrow |g|$
  $min \leftarrow \infty$, $i_{min} = 0$, $j_{min} = 0$, $A \leftarrow NULL$
  while $B_c \neq B$ **do**
    for $i = 1$ to B_c **do**
      for $j = i$ to B_c **do**
        $A \leftarrow LCA[g[i], g[j]]$
        if $min \leq \mathrm{P}(i, j)$ **then**
          $min \leftarrow \mathrm{P}(i, j)$, $i_{min} \leftarrow i$, $j_{min} \leftarrow j$
    $g \leftarrow (g - \{i_{min}, j_{min}\}) \cup A$
    Increase BAT of A by BAT of i_{min} and j_{min}.
    $B_c \leftarrow B_c - 1$
  return g
Procedure: P(i, j)
  return $\mathbb{N}[i].BAT \times (\mathrm{dist}[i, N] - \mathrm{dist}[N, LCA[i, j]]) + \mathbb{N}[j].BAT \times (\mathrm{dist}[j, N] - \mathrm{dist}[N, LCA[i, j]])$

Theorem 1. *The complexity of the First Greedy Blocking Strategy is* $O(NB)$.

Proof. If there are N nodes and B is the budget, then the algorithm would add a node to assignment set for $B-1$ times. After each iteration, it takes $O(N)$ time to update weight. Therefore, the complexity of the strategy is $O(NB)$. In the worst case $B = N$ and the complexity is $O(N^2)$. □

C. Second Greedy Solution

We start with the maximum number of filters needed to block attack traffic. In fact, we are assigning filters to all non-white entry nodes initially. Then gradually the number of filters and selected FRs will be reduced by one by merging a couple of filters. The merged filter will be sent to the least common ancestor (LCA) of the two FRs. Merging two filters means simply adding the conditions by "or" relation. There could be many options to merge two filters and merging will be associated with penalty. When we are merging two filters of two FRs and assigning the new filter to LCA of them, we are yielding attack traffic to the LCA FR. The amount of attack traffic we are yielding is the penalty associated with the merge. We select a couple of FRs with lowest merging penalty. Thus the number of selected FRs or filters is reduced by one. The reduction process continues until it meets the budget. The algorithm is shown in Alg. 1.

Let us consider the tree in Fig. 3(a). We use a double linked tree data structure. Each node contains a pointer to its parent, a number representing blocked attack traffic (BAT), an array of pointer to children with distance, and the color of the node. Initially the BAT of a "white" node is 0. The BAT of a "black" entry node is the total attack traffic. The BAT of non-entry nodes is 0. We keep an array $\mathbb{N}$ of pointers to the nodes to quickly access a node by its label. $\mathbb{N}[i]$ is the node having level i. Firstly, we need to simplify the tree. There is only one node 6 without fork. We remove the node 6 and make 4 child of its parent 7. The new distance to 4 from 7 will increase by the distance of deleted link. The deletion of a node can be done in constant time. Finding out all non-forked nodes takes $O(N)$ time. Therefore, the simplification can be done in $O(N)$ time. Next steps are taken according to the Alg. 1. B_c is the number of non-white entry nodes ($B_c = 3$). The assignment set $g = \{1, 2, 4\}$. This step takes $O(N)$ time to find all the entry nodes. Then the all-pair LCA is computed. LCA calculation can be represented as the following recurrence:

$$LCA[i,j] = \begin{cases} i, & \text{for } i = j \\ LCA[i, p(j)], & \text{for } LCA[i, p(j)] \neq null \\ LCA[p(i), j], & \text{for } LCA[p(i), j] \neq null \\ LCA[p(i), p(j)], & \text{otherwise} \end{cases} \quad (4)$$

Here $p(i)$ represents the parent of node i which can be found in constant time. The all pair LCA can be calculated in $O(N^2)$ time if we use dynamic programming. We have $LCA[i, j]$ is calculated in a top-down fashion. Therefore, $LCA[7, 7]$ is calculated first and $LCA[7, 7] = 7$. After that $LCA[7, 5]$ is calculated. As $P(5) = 7$ so that $LCA[7, 5] = 7$. When we calculate $LCA[4, 3]$, we need to look at $LCA[7, 3]$, $LCA[5, 4]$, or $LCA[7, 5]$. The $LCA[7, 3] = 7$ so $LCA[4, 3] = 7$. Similarly, we calculate the rest of the pairs (see Fig. 3(b)).

Next we calculate distance ($dist$) of every node from the root. This calculation takes $O(N)$ as it needs to traverse the whole tree once again. The complete $dist[i, j]$ is shown in Fig. 3(b). When $B_c = 3$ the filter assignment is $\{1, 2, 4\}$. When $B_c = 2$, we have three options for merging filters: (1) merge filters of 1 and 2 and assign the merged filter to FR 5 ($P(1, 2) = 19$), (2) merge filters of 1 and 4 and assign the merged filter to FR 7 ($P(1, 4) = 14$), and (3) merge filters of 2 and 4 and assign the merged filter to FR 6 ($P(2, 4) = 36$). Therefore, option (2) is chosen and the assignment is $\{2, 7\}$ for $B_c = 2$. When $B_c = 1$, we have one choice which is to merge filters of 2 and 4 and assign the merged filter to 7 ($P(2, 7) = 30$). Therefore, for $B = 1$ the assignment is $\{7\}$.

Theorem 2. *The complexity of the Second Greedy Blocking Strategy is* $O(N^2(N - B))$.

Proof. The algorithm would iterate the step 5 loop for $N -$

Algorithm 2 DP Blocking Strategy

Input: The number of filters B, topology tree T.
Output: A set of nodes in T.

Procedure: BLOCK-DP(B, T)
  $N \leftarrow$ number of nodes in T
  for $i = 1$ to N **do**
    for $j = 0$ to B **do**
      if i is an entry node **then**
        Initialize $A[i,j]$, $L[i]$, $C[i]$, and $R[i,j,k]$
      else
        $L[i] \leftarrow \sum_{k=1}^{\Delta} L[c_k(i)]$
        $min \leftarrow= \infty$
        if No attacker attached to i **then**
          $p \leftarrow$ cost according to Equation 6
          if $min \geq p$ **then**
            $min \leftarrow \sum_{k=1}^{\Delta} A[c_k(i), x_k]$
            $A[i,j] \leftarrow min$
            $\forall_{1 \geq k \geq \Delta} R[i,j,k] \leftarrow x_k$,
            $R[i,j,\Delta+1] \leftarrow 1$
        for $\forall x_1, x_2, ..., x_k : \sum_{k=0}^{\Delta} x_k = j$ **do**
          $p \leftarrow \sum_{k=1}^{\Delta} A[c_k(i), x_k]$
          if $min \geq p$ **then**
            $A[i,j] \leftarrow p \leftarrow min$
            $\forall_{1 \geq k \geq \Delta} R[i,j,k] \leftarrow x_k$
            $R[i,j,\Delta+1] \leftarrow 0$
  return FIND-ASSIGNMENT(R, B)

B times. In each iteration, it takes $O(N^2)$ time to find out the pair of nodes with minimum penalty. The all pair LCA computation takes $O(N^2)$ time. The complexity of Alg. 1 is $O(N^2(N-B)) + O(N^2) = O(N^2(N-B))$. In the worst case $B = 1$ and the complexity is $O(N^3)$. The complexity can further be improved to $O((N-B)^2 \log N)$ using min-heap data structure (see Appendix A for details). □

V. Destination-based Filter Assignment Policy

As we are using destination-based filters for protection against spoofed DDoS attack, we are blocking some LUs. In this section, we formulate another optimization problem of assigning destination-based filters to the FRs so that a weighted sum of the contamination and blocked LUs is minimum.

A. *Problem 2: Find a filter assignment so that the LU blockage and contamination are minimal.*

It is always better if the victim can select some FRs within its budget which minimizes both the number of blocked LUs and contamination. As discussed in Section III the source-based filter cannot ensure protection against IP spoofing DDoS attack. For example, if the attacker attached to node 1 uses IP address of the users attached to node 3 (see Fig. 3(a)). The filter used at node 1 or 5 would forward the packet. But if the FRs use destination-based filter then no spoofed attack packet can penetrate. Therefore, the victim would use the destination-based filters. The problem can be expressed as the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \omega W_c(g) + (1-\omega)|U_b(g)| \\ \text{subject to} \quad & |g| \leq B, \forall g \subset G, v \notin G \end{aligned} \tag{5}$$

Here $\omega = [0,1]$ is considered a system parameter which determines priority of total contamination and LU blockage.

Algorithm 3 Find Assignment

Procedure: FIND-ASSIGNMENT(R, B)
  $x.i \leftarrow N, x.j \leftarrow B$, $g \leftarrow \emptyset$, and $Q \leftarrow \emptyset$.
  ENQUEUE(Q,X).
  while $Q \neq \emptyset$ **do**
    $x \leftarrow$ DEQUEUE(Q)
    if $R[x.i, x.j, \Delta+1] \neq 0$ **then**
      $g \leftarrow$ the $R[x.i, x.j, \Delta+1]$ nodes according to case 1.
    else
      for $k = 1$ to Δ **do**
        $x_c.i \leftarrow c_k(x.i)$, $x_c.j \leftarrow R[x.i, x.j, k]$
        ENQUEUE(Q, x_c)
  return g

B. A Dynamic Programming Solution

Let us consider an N node tree with maximum node degree Δ. The nodes are labeled in bottom-up and left-right order. We define A as a $N \times B$ array which contains optimal cost for every node and budget. For example $A[i,j]$ is optimal cost of budget j on subtree rooted by node i.

We define L as a $1 \times N$ array which contains the number of LUs in subtree rooted by every node. $L[i]$ is the number of LUs in subtree rooted by node i. We also define R as an $N \times B \times (\Delta+1)$ array which contains the number of filters assigned to node i and its subtrees for every node and budget. For example, $R[i,j,1]$, $R[i,j,2]$, and $R[i,j,\Delta+1]$ are the number of filters to first subtree, second subtree, and node i of subtree rooted by i for budget j. The optimal cost of using j destination-based blocking/filter in subtree rooted by i is the minimum of the following quantity:

Case I: The minimum total weighted cost, if we assign 1 filter to the node i and the rest of the filters to some nodes of the subtree rooted by i. Therefore, the cost will be a weighted sum of the minimum contamination and LU in subtree rooted by i. The assignment of the $j-1$ nodes can be done in a greedy way. We can apply the Alg. 1 to find an assignment in subtree rooted by i but the cost will no longer be optimal. First, we assume an attacker attached to the i. This assumption confines a filter to i. Then we find an assignment of budget j using Alg. 1. As i will be assigned a filter, the other $j-1$ filters will be assigned to the subtree rooted by i.

Let $g'[i, j-1]$ be the assignment which provides contamination $(C(g'[i,j-1]))$ to the subtree rooted by i for $j-1$ filters. Then the cost for this option will be:

$$A[i,j] = \omega C(g'[i,j-1]) + (i-\omega)L[i] \tag{6}$$

Case II: The minimum total weighted cost, if we divide the number of filters into $x_1, x_2, ..., x_k$ parts and assign them to the subtrees $c_1, c_2, ..., c_k$, respectively. Therefore, the cost for this option will be:

$$A[i,j] = \sum_{k=1}^{\Delta} A[c_k, x_k] \tag{7}$$

Therefore, we take the minimum quantity from the above two options. If there are some attackers attached to node i, we do not consider the option 2. This is because, if we assign all the j filters to its subtree then the attack traffic from i will reach the victim v, which is not allowed by the constraint of the problem definition.

A

i \ j	0	1	2	3
1	∞	0.5	0.5	0.5
2	∞	1	1	2
3	∞	0	0	0
4	∞	1	1	1
5	∞	2	2	2
6	∞	14	6.5	1.5
7	∞	7.5	1.5	1.5
8	∞	36.5	21.5	14

R

i \ j	0	1	2	3
1	0,0,0,0	0,0,0,1	0,0,0,2	0,0,0,3
2	0,0,0,0	0,0,0,1	0,0,0,2	0,0,0,3
3	0,0,0,0	0,0,0,1	0,0,0,2	0,0,0,3
4	0,0,0,0	0,0,0,1	0,0,0,2	0,0,0,3
5	0,0,0,0	0,0,0,1	0,0,0,2	0,0,0,3
6	0,0,0,0	0,0,0,1	0,0,0,2	1,1,1,0
7	0,0,0,0	0,0,0,1	1,1,0,0	2,1,0,0
8	0,0,0,0	0,0,0,1	1,1,0,0	2,1,0,0

L

i	
1	1
2	2
3	0
4	1
5	2
6	3
7	10
8	23

Fig. 4. A, R, L, and C.

Let us consider the traffic topology in Fig. 3(c) and $\omega = 0.5$. The leaf entry nodes are $1, 2, 3, 4,$ and 5. The calculations of A, R, and L are straightforward. For example $A[1,0] = \infty$, $A[1,1] = 0.5 \times 0 + 0.5 \times 1 = 0.5$, and $A[1,2] = 0.5 \times 0 + 0.5 \times 1 = 0.5$. $R[i,1] = [0,0,0,1]$, $R[i,2] = [0,0,0,2]$, and $R[i,3] = [0,0,0,3]$. For node 6 and $j = 0$, we have one choice $x_1 = 0, x_2 = 0, x_3 = 0$ and the cost is ∞. For $j = 1$, we have two cases. **Case 1:** 1 for node 6, and $x_1 = 0, x_2 = 0, x_3 = 0$. The total cost is $0.5 \times 25 + 0.5 \times 3 = 14$. **Case 2:** 0 for node 6, and (1) $x_1 = 1, x_2 = 0, x_3 = 0$, (2) $x_1 = 0, x_2 = 1, x_3 = 0$, or (3) $x_1 = 0, x_2 = 0, x_3 = 1$. For option (1): total cost is $\infty + \infty + \infty = \infty$. For option (2) and (3) total cost is also ∞. Therefore, case 1 is minimum ($A[6,1] = 14$) and $R[6,1] = [0,0,0,1]$.

For $j = 2$, there are also two cases. **Case 1:** 1 for node 6 and 1 is for its subtrees. We assume an attacker attached to 6. Then applying the Alg. 1 for $B = 2$, we find the assignment is $\{6, 2\}$. After assigning the filters the total contamination is $4+6 = 10$ The total cost in this option is $10(0.5)+3(1-0.5) = 6.5$. **Case 1:** 0 for node 6, and 2 filters to subtrees of 6. This case is valid for node 6 because there is no attacker directly attached to it. There can be six options: (1) $x_1 = 2, x_2 = 0, x_3 = 0$, (2) $x_1 = 0, x_2 = 2, x_3 = 0$, (3) $x_1 = 0, x_2 = 0, x_3 = 2$, (4) $x_1 = 0, x_2 = 1, x_3 = 1$, (5) $x_1 = 1, x_2 = 0, x_3 = 1$, and (6) $x_1 = 1, x_2 = 1, x_3 = 0$. For option (1), the total cost is $A[1,2] + A[2,0] + A[3,0] = 0.5 + \infty + \infty = \infty$. Similarly, the options (2) to (6) cost ∞. Therefore, case 1 is minimum and $A[6,2] = 6.5$ and $R[6,2] = [0,0,0,2]$. Similarly, we calculate the rest of the entries in A and R. The complete A, L and R are shown in Fig. 4.

According to the definition, $A[8,3]$ contains the cost for budget $B = 3$ which is 14. From R we can find out which FRs are blocked. $R[8,3,1] = 2$ and $R[8,3,2] = 1$ means 2 and 1 filters are assigned to its 1st and 2nd subtrees, respectively. Then we need to look $R[6,2,k]$ and $R[7,1,k]$. $R[6,2,1] = 0$, $R[6,2,2] = 0$, $R[6,2,3] = 0$, and $R[6,2,4] = 2$ means no filter is assigned to its subtrees and two filters are assigned to itself. Therefore, we need to find the assignment according to Case I. According to Case I $\{6, 2\}$ is the assignment. Similarly, we can find that a filter is assigned to node 7. So, the filter assignment is $\{2, 6, 7\}$ for budget $B = 3$ and the cost is 14.

Theorem 3. *Complexity and space needed of the DP Blocking Strategy are $O(NB^{\Delta-1})$ and $O(NB\Delta)$.*

Proof. Let us consider the topology is a N node tree with maximum node degree Δ and the victim has budget of B. To find the partitions $x_1, x_2, ..., x_\Delta$ we need $O(B^{(\Delta-1)})$ time if we use naive nested iteration approach. Therefore, the complexity of the Alg. 2 is $O(NB^{(\Delta-1)})$. The total space needed is $(N-1)B+(\Delta+1)(N-1)B+(N-1)$ which is an order of $O(NB\Delta)$. For a binary tree topology the complexity is $O(NB^2)$ and the space complexity is $O(NB)$. □

Theorem 4. *The Alg. 2 provides optimal solution when $\omega = 0$.*

Proof. When $\omega = 0$ only blockage of LUs is taken into account. The Alg. 2 uses a dynamic programming bottom-up strategy to search the optimal assignment. For an one-node tree, if the node color is "black" or "gray" then there is no solution for $B = 0$. Because without any filter, the attack traffic will be forwarded to the downstream routers. For $B \geq 1$ there is only one choice of selecting FR which is that node. If that node is selected, the optimal number of blocked LUs is the LUs attached to it. In each step, the Alg. 2 chooses the best allocation of filters to itself, left subtree, or right subtree. Therefore, the Alg. 2 provides optimal filter assignment to the FRs through exhaustive search. □

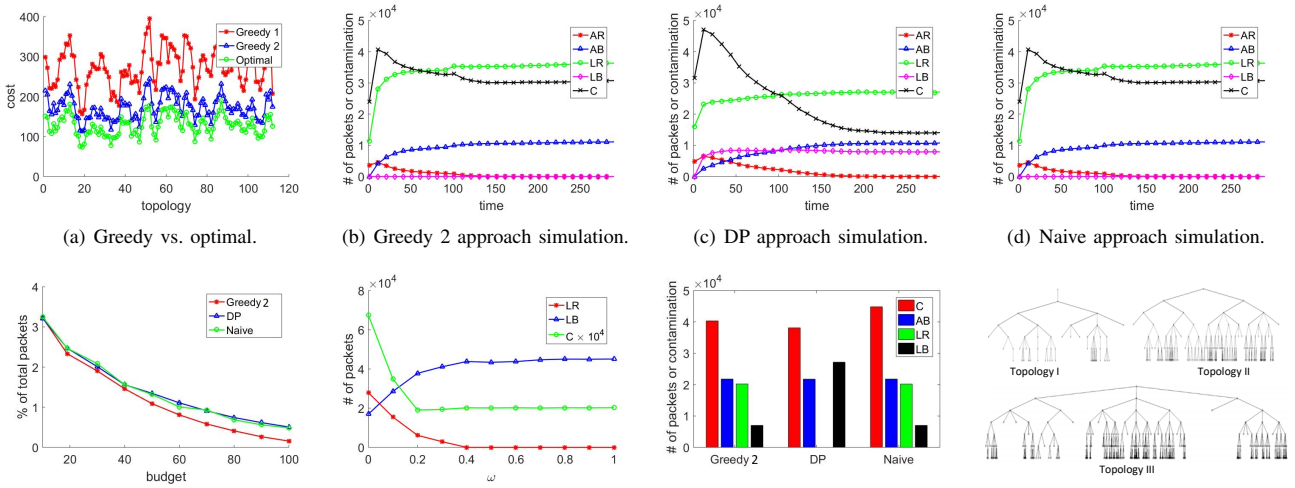
VI. Experimental Results

A. Experimental Setting

We conducted the experiments with a custom build java simulator. The main reason for using custom build for simulator is its scalability. We do not need to analyze transmission time, bandwidth, or packet drop issues. We only need to count the number of legitimate (or attack) received (or blocked) packets. The network topologies we considered contain $100 - 500$ routers. Using NS3 or other similar simulator 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 simulation for randomly generated tree topology and a subset from a real network topology. We used two randomly generated topology having node degree between $[0 - 4]$, internal node user probability between $1 - 0.25$, and maximum depth 6. The entry nodes' color and number of users or attackers were selected randomly from a uniform distribution. The Topology I and II were randomly generated trees with 66 and 247 nodes and max node degree 4. The Topology III was taken from a subset of the Stanford University Note Dame web graph [13]. The dataset contained $325,729$ nodes and we took a subset (which is a tree) containing 403 nodes. Then we randomly assigned users to the tree with internal user probability 0.1. The details are shown in Table I and Fig 5(h).

We compare the performances of our proposed second greedy blocking strategy (Greedy 2), DP blocking strategy (DP) and a naive approach. In the naive approach, source-based filters are used. At first, a filter is assigned to the node which is attached to v. Then the filter is split to its child nodes which have higher attack traffic flows. The split only occurs if the node does not have any attacker attached to it. Therefore, a split will increase the number of filters by its number of children. Every round the node with highest attack traffic is



(a) Greedy vs. optimal. (b) Greedy 2 approach simulation. (c) DP approach simulation. (d) Naive approach simulation.

(e) Total cost of three solutions. (f) DP effect of ω. (g) Simulation with real topology. (h) Traffic topologies.

Fig. 5. Simulation results.

TABLE I
TOPOLOGY PARAMETERS

	Topology I	Topology II	Topology III
Number of nodes	66	247	403
Internal user probability	0.1	0.25	0.1
Attacker ratio	0.4	0.4	0.4
Max Node Degree	4	4	20
Data Rate(pack/ms)	[0.1-0.4]	[0.1-0.4]	[0.03-0.13]

selected for split from the current assignment. The process continues until the number of assigned filters is less than budget. We show the contamination (C), number of blocked attack packets (AB), number of received attack packets (AR), number of blocked legitimate packets (LB), and number of received legitimate packets (LR) for the three topologies and three approaches.

B. Simulation Results

We compare the performances of the greedy solutions for different randomly generated topologies. The Fig. 5(a) shows the cost of first greedy (Greedy 1), second greedy (Greedy 2), and optimal cos. For $[10, 15]$ node trees and $[3, 5]$ filters the Greedy 2's average cost is about 9% higher than average optimal cost. For $[25, 35]$ node trees and $[5, 10]$ filters the Greedy 2's average cost is about 24% higher than optimal. Though the complexity of Greedy 2 is little higher than Greedy 1 it performs much better.

Figs. 5(b) , 5(c), and 5(d) show the C, AB, AR, LB, and LR for the Topology I. Here the contamination is the number of attack-packet forwarding events per attack packet. We can see that, at the beginning, the C in every approach is higher. The C reduces over time and becomes gradually more stable. This is because, at the beginning the victim knows a small subset of the complete topology. Over time, the victim gets more and more information from the marked packet and reconstructs the traffic topology. Finally, the victim succeeds in constructing the complete topology. That is why the AR is high at the beginning and decreases over time and finally converges to zero. The AB shows the opposite behavior for the same reason. We can also observe that the cost (contamination) of second greedy solution is less than the naive approach. The cost of DP is a weighted sum of C and BL.

Next we use the Topology II to compare the performances of three approaches for different budgets after convergence (considering the victim knows complete topology). In this topology we increased the internal user probability to see the amplified effect of different budget settings. Each run was observed for longer time (200-400 slots and each slot is about 1 second long) and the average measurements of slots are plotted. Fig. 5(e) shows the cost of the three approaches. Here, the cost is the number of forwarding events per attack traffic. We can see that the cost of Greedy 2 is then lowest. The costs of DP and Naive are similar.

Fig. 5(f) shows the effect of ω in the DP. We observe the C, RL, and BL by ω. The C is multiplied by $10,000$ to show the effect clearly with the other measurements. If we set $\omega = 0$, the total RL and C are highest (because we give no weight to contamination). On the other hand, the LB is lowest at $\omega = 0$. When $\omega \geq 0.4$ all the legitimate packets are blocked and the contamination remains stable. With the increase of ω, the LB increases and the LR and C decrease, which means that the more (or less) weight we put to contamination the more (or less) the legitimate traffic we block. Therefore, there is a trade-off between the number of legitimate traffic and attack traffic. The victim should set a proper ω to have a good balance between blocked legitimate packets and contamination.

Fig. 5(g) shows a comparison of C, AB, LR, and LB using a real topology. Here, we define C as the number of forwarding events of attack packets. We use the Topology III for this simulation. The simulation shows that the C of Greedy 2 is lower than Naive approach. In Greedy 2 and Naive approaches attack packets are forwarded $40,330$ and $44,844$ on average. The AB, LB, and LR are similar for both approaches. The DP shows the lowest contamination $(38, 126)$ but highest blocked legitimate traffic (for $\omega = 0.5$). The number of blocked LUs can be adjusted by choosing a lower ω.

VII. Conclusion

The DDoS attack is the most powerful attack to make a service unavailable to users. It is not possible to protect any server from DDoS attack without the help of the network equipment. As the most important component in a network, routers can be upgraded to filter routers easily. Besides, the filter router can work in a network with legacy routers. In the four-phase DDoS protection system, the filter routers block the attack traffic according to the victim's instruction. Though the blocking control of an internet service provider (ISP) is at victims hand who may not belong to the ISP but it will help the ISP to minimize traffic congestion. Therefore, both parties are benefited. In this work, we present three filter scheduling policies for two different settings. We compare the performances of proposed scheduling policies with the naive approach. Simulation results show that our proposed approaches work better than the naive approach. Both the source-based and destination-based filters have some advantages and limitations. In future, we may formulate another problem for finding an optimal assignment using the filter type most fitted to a filter router.

Acknowledgments

This research was supported in part by NSF grants CNS 1757533, CNS1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS1460971, and IIP 1439672.

References

[1] United States Code: Title 18,1030, "Fraud and related activity in connection with computers — government printing office," http://www.gpo.gov, 2014.

[2] "Cloudflare," https://blog.cloudflare.com/.

[3] *Attack Types*. Wiley-Blackwell, 2017, ch. 5, pp. 113–141.

[4] J. Wang and I. C. Paschalidis, "Statistical traffic anomaly detection in time-varying communication networks," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 2, pp. 100–111, June 2015.

[5] V. Matta, M. D. Mauro, and M. Longo, "Ddos attacks with randomized traffic innovation: Botnet identification challenges and strategies," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1844–1859, Aug 2017.

[6] H. Luo, Z. Chen, J. Li, and A. V. Vasilakos, "Preventing distributed denial-of-service flooding attacks with dynamic path identifiers," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1801–1815, Aug 2017.

[7] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Y. Yau, and J. Wu, "Realtime ddos defense using cots sdn switches via adaptive correlation analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 7, pp. 1838–1853, July 2018.

[8] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and M. Rajarajan, "Scale inside-out: Rapid mitigation of cloud ddos attacks," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2017.

[9] C. Wang, T. T. N. Miu, X. Luo, and J. Wang, "Skyshield: A sketch-based defense system against application layer ddos attacks," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 559–573, March 2018.

[10] B. Rashidi, C. Fung, and E. Bertino, "A collaborative ddos defence framework using network function virtualization," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2483–2497, Oct 2017.

[11] D. Seo, H. Lee, and A. Perrig, "Pfs: Probabilistic filter scheduling against distributed denial-of-service attacks," in *2011 IEEE 36th Conference on Local Computer Networks*, Oct 2011, pp. 9–17.

[12] D. Seo, H. Lee, and A. Perrig, "Apfs: Adaptive probabilistic filter scheduling against distributed denial-of-service attacks," *Comput. Secur.*, vol. 39, pp. 366–385, Nov. 2013.

Algorithm 4 $O((N-B)^2 \log N)$ Greedy Blocking Strategy

Procedure: BLOCK-GREEDY-2(B, T)
  Steps $1-6$ in Alg. 1
  $S \leftarrow \{x : x.i, x.j \in g, i \neq j,$ AND $x.key = \mathrm{P}(x.i, x.j)\}$
  Create minheap H from S.
  while $B_c \neq B$ **do**
    $m \leftarrow$ ExtractMin(H)
    Remove x form H, where $x.i$ or $x.j$ is $m.i$ or $m.j$
    $g \leftarrow g - \{i, j\}$
    Insert all $\{x : x.i = LCA[m.i, m.j], x.j \in g\, x.key = \mathrm{P}(x.i, x.j)\}$
    $g \leftarrow g \cap \{LCA[m.i, m.j]\}$
    Increase BAT of A by BAT of i_{min} and j_{min}.
    $B_c \leftarrow B_c - 1$
  return g

[13] "Note dame web graph," https://snap.stanford.edu/data/web-NotreDame.html.

Appendix

A. $O((N-B)^2 \log N)$ approach for Second Greedy Solution

In the Alg. 4, Steps 1-6 are the same as in Alg. 1. Instead of searching for minimum penalty pair, we create a min-heap H from all pairs of elements in the current assignment g. Then, we do ExtractMin(H) from the heap and remove all associated elements with the removed nodes. Then, we insert all the associated elements with the new node. We create a min-heap of all the pairs of nodes in g (according to Fig. 3(a)). The heap H contains $\binom{N}{2}$ elements. The key of an element (i, j) is the penalty of merging the pair of nodes $P(i, j)$. For example, the key of node $(1, 2)$ is $P(1, 2) = \mathbb{N}[1].BAT \times (\mathrm{dist}[1, 7] - \mathrm{dist}[5, 7]) + \mathbb{N}[2].BAT \times (\mathrm{dist}[2, 7] - \mathrm{dist}[5, 7]) = 4 \times (2-1) + 15 \times (2-1) = 19$. Similarly, the key of node $(1, 4)$ and $(2, 4)$ is $P(1, 4) = 14$ and $P(2, 4) = 36$. Therefore, the heap H will have $\{(1, 4), (1, 2), (2, 4)\}$ elements. The heap construction takes $O(N^2)$ time for $\binom{N}{2}$ elements.

The next step is a while loop which iterates until B_c is not equal to the budget B of v. The ExtractMin (H) will return the $(1, 4)$ because its key is lowest. Generally, the ExtractMin(H) operation on heap takes $O(\log n)$ time. Here we have $\binom{N}{2}$ elements which take $O(\log N^2) = O(2 \log N) = O(\log N)$ time. After that we remove elements of heap which are associated with 1 or 4. Therefore, the heap H contains no elements because all the elements are associated with 1 or 4. The number of remove operations is $O(N)$ and each remove operation takes $O(\log N)$. Now we remove 1 and 4 from g. Therefore, $g = \{2\}$. Next we add each pair of elements associated with $LCA[4, 1] = 7$ and g. So, element $(7, 2)$ will be added to H. At this step we calculate the key of $(7, 2) = (15 \times 2 + 0) = 30$. The BAT of node 7 will be increased by $P(1, 4) = 14$. So, BAT of 7 is $(0 + 14) = 14$. Then $LCA[1, 4] = 7$ is added to g and B_c is decreased by 1. As the $B_c = 2$ is now equal to B the loop stops and our algorithm finishes by returning $g = \{2, 7\}$.

We can see that in the heap an element (a pair of nodes) is inserted or deleted once. The total number of removals or insertions is $\binom{N-B}{2}$. Each insert or delete operation takes $O(\log((N-B)^2)) = O \log(N)$. Therefore, the complexity of the algorithm is $O((N-B)^2 \log(N))$.