Source Address Filtering in Large Scale Network: A Cooperative Software Mechanism Design

Shu Yang[*], Mingwei Xu[*], Dan Wang[†], Jianping Wu[*]
[*]Dept. of Computer Science and Technology, Tsinghua University
[†]Dept. of Computing, Hong Kong Polytechnic University

1. INTRODUCTION

To prevent network infrastructure from malicious traffic, such as DDoS attack and scanning, source filtering is widely used in the network. There are different ways to store the filters, e.g., a blacklist of source addresses. Among them, TCAM-based is used as the de facto, because of its wire speed performance. Unfortunately, TCAM is a scarce resource because it's limited by small capacity, high power consumption and high cost. To save storage space, some TCAM-based solutions even block part of the legitimate traffic for better aggregation. Another choice is software based solutions, which have larger storage space compared to hardware based solutions. However, they require multiple accesses for a single lookup, which causes latency.

Traditional schemes place the filter at border routers, where transit traffic is certain to pass by. This largely increases processing burden on border routers, while routers at other locations still have spare resources.

In this paper, we design a new mechanism for source filtering. We store the filters in software. Each router just has to lookup a few bits in the source addresses, and routers along the forwarding path of a packet cooperatively lookup all bits in the source addresses. Thus we can both reduce the processing time and block all malicious traffic with enough memory space. Besides, we want to balance the load across different routers.

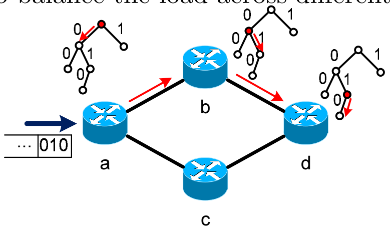

Figure 1: Example of our mechanism for source filtering

We first use a simple example to explain our idea. In Figure 1, router a is the ingress and d is the outgress router. Assume the source address has 3 bits, and there are 3 source filters: 1*, 00*, 010. The source filters are organized as uni-bit tries. Traditionally, the filters are placed at border router, e.g., router a. Thus a may require 3 accesses to memory to filter malicious traffic in the worst case.

In our new mechanism, router a has only to check the 0_{th} bit, b, c check the 1_{st}, and d checks the 2_{nd}. Assume that a packet with source address 010 arrives at router a, and the path towards destination is $\{a, b, d\}$. First, router a checks the 0_{th} bit in source address, and moves the pointer from the root to the 1_{st} level, then it passes the packet to router b. b checks the 1_{st} bit and passes the packet to d, which checks the 2_{nd} bit and concludes that the packet should be filtered. With the mechanism, each router requires only one access to memory. When the network is large, the amortized burden on each router can be quite low.

Due to page limit, protocol design is out of our consideration. In this paper, we formulate the problem as finding a scheme such that 1) along any path that a packet may travel through, all bits in source address should be *covered*, i.e., be checked by a router along the path and 2) the load should be balanced across different routers. We show that the problem is NP-complete and develop a heuristic algorithm to solve it.

At last, we conduct a case study using the topology of China Education and Research Network 2 (CERNET2), the world's largest IPv6 backbone network (including 59 Giga-PoPs). The results show that the load is better balanced with the new mechanism.

2. PROBLEM FORMULATION

Let $G = (V, E)$ be a network, where V is the set of routers, and E is the set of links. Let $\mathcal{R}$ denote the set of border routers in the network. Let $\mathcal{L}$ be a path (i.e., an ordered set of routers), $\mathcal{P}$ be the set of paths that a packet may traverse through the network. For $u, v \in \mathcal{L}$, define $u \preceq_{\mathcal{L}} v$ as node u be the predecessor of v on $\mathcal{L}$.

Let $\mathcal{T} = \{b_0, b_1, \ldots, \}$ $(0 \leq b_i \leq 31$ for IPv4, and $0 \leq b_i \leq 127$ for IPv6) be the set of bits that has to be checked. Each router has only to check a few bits in source addresses, let $\mathcal{B}_v \subseteq \mathcal{T}$ be the set of bits that v should check, and $\overrightarrow{B} = (\mathcal{B}_{v_1}, \mathcal{B}_{v_2}, \ldots), v_i \in V$ be a vector representing a *covering scheme*. Along a path, all routers cooperatively check all bits in $\mathcal{T}$ from higher to lower bits. Each router in the network has limited capacity due to CPU limitations. We model this as the maximum number of additional bits in source addresses that the router can process. Let C_v be the capacity of v. To balance the load across the network, let $f(x)$ be a piecewise-linear increasing, convex function, and $F(\overrightarrow{B}) = \sum_v f(|\mathcal{B}_v|/C_v)$ be the total cost of $\overrightarrow{B}'$. Then we can formulate the problem as follows,

$$\min F(\overrightarrow{B}) \qquad (1)$$

$$\text{s.t. } |\mathcal{B}_v| \leq C_v, \forall v \in V \qquad (2)$$

$$\bigcup_{v \in \mathcal{L}} \mathcal{B}_v \supseteq \mathcal{T}, \forall \mathcal{L} \in \mathcal{P} \qquad (3)$$

$$\bigcup_{u \preceq_{\mathcal{L}} v} \mathcal{B}_u = \mathcal{T} \text{ or } \bigcup_{u \preceq_{\mathcal{L}} v} \mathcal{B}_u = \{p \in \mathcal{T} | p < q, \forall q \in \mathcal{B}_v\}, \forall v \qquad (4)$$

Eq. (1) expresses the objective to minimize the total cost on all routers. Eq. (2) expresses the constraint on capacity. Eq. (3) states that all bits must be covered along any path that a packet can traverse. Eq. (4) states that on a router which is on the path from ingress to outgress, if not all bits have been checked, then the successor node should check lower bits. We call the solution to the problem the optimal covering scheme.

3. OPTIMAL COVERING SCHEME

THEOREM 1. *Finding the optimal covering scheme is NP-complete.*

PROOF. Due to page limit, we put the proof in [3]. □

Next, we develop Algorithm Opt-Cover() based on tabu-search, which is widely used to solve optimization problems in many applications[1].

We call $\overrightarrow{B}' = (\mathcal{B}'_{v_1}, \mathcal{B}'_{v_2}, \ldots)$ be a *neighbor* of $\overrightarrow{B}$, if $\exists v_i, v_j \in V, \exists p \in \mathcal{B}_{v_j}$ such that 1) $\mathcal{B}'_{v_i} = \mathcal{B}_{v_i} \cup \{p\}, \mathcal{B}'_{v_j} = \mathcal{B}_{v_j} \backslash \{p\}$ and 2) $\mathcal{B}'_{v_k} = \mathcal{B}_{v_k}, \forall k \neq i, j$ and 3) $\overrightarrow{B}'$ satisfies Eq. (2)-Eq.(4). Let $swap(v_i, v_j, p)$ be a swapping operation that adds p to $\mathcal{B}_{v_i}$ and deletes p from $\mathcal{B}_{v_j}$, For a covering scheme $\overrightarrow{B}$ and its neighbor $\overrightarrow{B}'$, we can find a swapping operation that transforms $\overrightarrow{B}$ to $\overrightarrow{B}'$, let $\mathcal{S}(\overrightarrow{B}, \overrightarrow{B}')$ denote the swapping operation. Besides, let $\mathbf{N}(\overrightarrow{B})$ be the set of all neighbors of $\overrightarrow{B}$.

Tabu-search iteratively searches from a covering scheme $\overrightarrow{B}$ to a new scheme $\overrightarrow{B}'$ in $\mathbf{N}(\overrightarrow{B})$. Opt-Cover() maintains a tabu list, that includes the recent swapping operations. Thus the neighboring space excludes $\overrightarrow{B}'$, where $\mathcal{S}(\overrightarrow{B}, \overrightarrow{B}')$ is in the tabu list. Opt-Cover() records the best covering scheme ever found, and stops if specified number of iterations happens since the last best covering scheme was found. Besides, Opt-Cover() permits overriding of tabu list when the covering scheme is better than the best covering scheme.

Algorithm 1: Opt-Cover()

```
Input   : $\mathcal{T}, \mathcal{P}$
Output  : $\mathcal{B}_v, \forall v \in V$
begin
    $\mathcal{B}_v \leftarrow \mathcal{T}, \forall v \in \mathcal{R}$, $\mathcal{B}_v \leftarrow \emptyset, \forall v \notin \mathcal{R}$
    $\overrightarrow{B} \leftarrow (\mathcal{B}_{v_1}, \mathcal{B}_{v_2}, \ldots)$, $tabu\_list \leftarrow \emptyset$, $\overrightarrow{Be} \leftarrow \overrightarrow{B}$
    while $Num <threshold$ do
        Find $\overrightarrow{B}' \in \mathbf{N}(\overrightarrow{B})$, where $\mathcal{S}(\overrightarrow{B}, \overrightarrow{B}') \notin tabu\_list$ and
        $F(\overrightarrow{B}')$ is minimized, $Num{+}{+}$
        if $\overrightarrow{B}'$ does not exist then
            Find $\overrightarrow{B}' \in \mathbf{N}(\overrightarrow{B})$, where $F(\overrightarrow{B}') < F(\overrightarrow{Be})$ and
            $F(\overrightarrow{B}')$ is minimized
            if $\overrightarrow{B}'$ doest not exist then return $\overrightarrow{B}$
        if $\overrightarrow{B}'$ is the best solution then $Num \leftarrow 0$, $\overrightarrow{Be} \leftarrow \overrightarrow{B}'$
        $tabu\_list = tabu\_list \cup \mathcal{S}(\overrightarrow{B}, \overrightarrow{B}')$
end
```

4. A CASE STUDY

We conduct a case study with the real topology of CERNET2. CERNET2 has two international exchange centers connecting to the Internet, Beijing (CNGI-6IX) and Shanghai (CNGI-SHIX). We want to block malicious traffic from CNGI-6IX to CNGI-SHIX along a predefined path that has six routers, i.e., {Beijing, Tianjin, Jinan, Hefei, Nanjing, Shanghai}. Figure 2 (the top bar) shows the estimated left capacity on each router. In this paper, we use the same piecewise linear function $f(x)$ with [2] ($f(x) = 5000$ if $x \geq 11/10$, $f(x) = 1$ if $x < 1/3$, and $f(x) = 3$ if $1/3 \leq x \leq 2/3$).

Traditional source filtering needs to check 128 bits (CERNET2 is an IPv6 network) in source addresses on border router. The additional burden exceeds the left capacity. The total cost is $\sum_v f(|\mathcal{B}_v|/C_v) = 5000$.

Figure 2 (the bottom bar) shows the results computed by Algorithm Opt-Cover(). Out of 128 bits, each router has only to check at most 47 bits in source addresses. The total cost on all routers is 8.

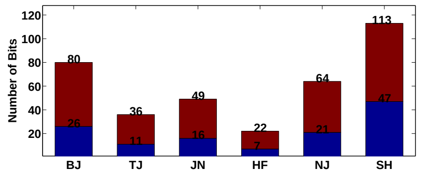

Figure 2: Number of additional bits that each router has to check in source addresses

5. ACKNOWLEDGMENTS

This work is supported by NSFC under grant No. 61073166 and 973 Program under grant No. 2009CB320502.

6. REFERENCES

[1] F. Glover. Tabu search – Part I. *ORSA J. on Computing*, 1(3):190–206, 1989.

[2] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On selfish routing in internet-like environments. In *Proc. ACM SIGCOMM'03*, Karlsruhe, Germany, Aug 2003.

[3] S. Yang and M. Xu. Source address filtering in large scale network: A cooperative software mechanism design. Technical report, Tsinghua University, Aug 2011. http://www.wdklife.com/tech-report.pdf.