Filtering Spoofed IP Traffic Using Switching ASICs

Jiasong Bai, Jun Bi, **Menghao Zhang**, Guanyu Li (Tsinghua University)
zhangmenghao0503@gmail.com

❖ *Motivation*: Could Hop-Count Filtering be improved?

Hop-Count Filtering (HCF)

- *Observation*: Although an attacker can forge any field in the IP header, she/he cannot falsify the number of hops an IP packet takes to reach its destination (determined by the Internet routing infrastructure).
- *Basic idea*: Filter spoofed IP traffic (e.g., TCP, UDP, ICMP) with an IP-to-hop-count (IP2HC) mapping table, which should only be updated by legitimate packets, i.e., TCP connections with established states.

Existing HCF enforcement mechanisms

- The monitoring of TCP establishment procedures can only be conducted in end hosts under traditional networks, these defense mechanisms are all located in end systems, which incurs bandwidth waste, additional latency and triangle routing.

Our Approach: *NetHCF*

- In-network spoofed IP traffic filtering system with switching ASICs
- Providing timely bandwidth protection for legitimate traffic, and saving computation and storage resources in end servers.
- Examining every packet to update the IP2HC table without sampling, and performing spoofed packets detection and filtering at line rate.

❖ *Challenges*

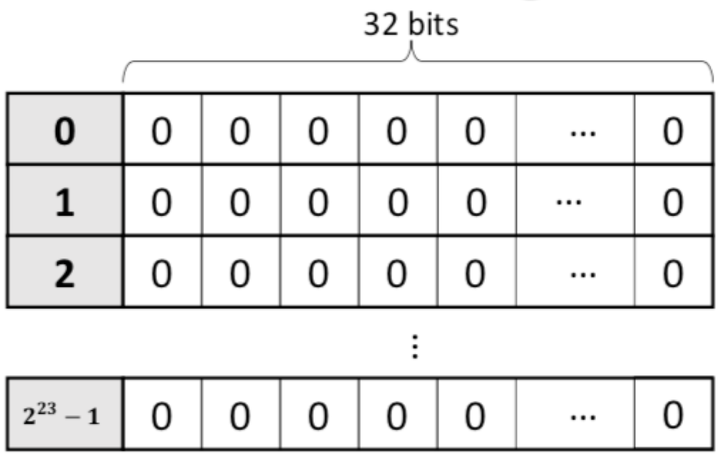
(1) Fitting the whole IP2HC table in limited SRAM and keeping the false positive rate acceptable

- ✓ It is impossible to store the IP2HC table at the granularity of each IP ad- dress (2^{32} bytes (4GB)) in limited switch SRAM (50~100MB).
- ✓ Given a specific storage size, the more mappings are added, the larger the false positive rate becomes as the probability of collision rises.

(2) Recording limited mappings and updating the IP2HC table timely

- ✓ With new mappings generated all the time, some mappings have to be removed timely, otherwise the false positive rate will increase.

❖ *NetHCF Design*



(a) Table Structure

Given an IP2HC table with m entries and k hash functions, after inserting n mappings ($n = \sum_{i=0}^{31} n_i$, n_i stands for number of packets with i hops), the probability that a specific bit i is still 0 is $p = (1 - \frac{1}{m})^{kn_i} \approx e^{-kn_i/m}$. The probability of a false positive is

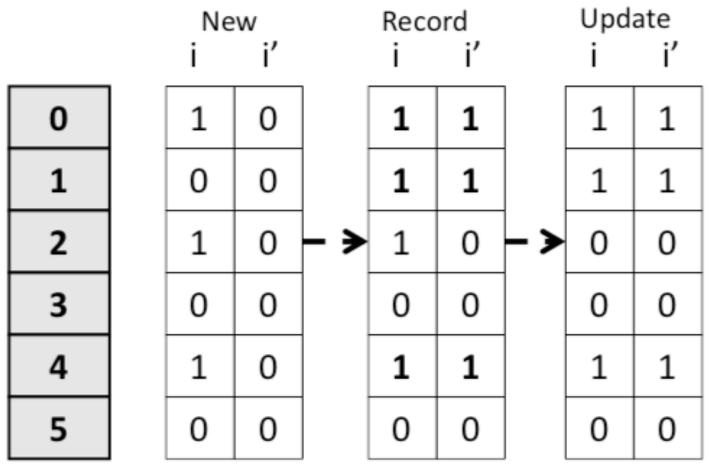
$$f = (1-p)^k = (1-e^{-kn_i/m})^k = e^{kln(1-e^{-kn_i/m})} = e^{\frac{-m}{n_i}ln(x)ln(1-x)}$$

f gets minimized when

$$x = 1 - e^{-kn_i/m} = \frac{1}{2} \Rightarrow k = \frac{m}{n_i}ln2 \Rightarrow f = (1/2)^{\frac{m}{n_i}ln2}$$

Set k as 7, we have $f < 0.01$ for $n_i < 8.5 \times 10^5$. Considering n_i is the number of mappings sharing the same hc value i, we have $n = 32 * \overline{n_i}$.

(b) Theoretical Analysis on False Positive



(c) Update Procedure

(1) The Structure of IP2HC Table

- ☐ IP2HC table compresses the address space from 2^{32} to 2^{23}, and each entry maintains a 32-bit bitmap to record all possible hop-count values. The i-th bit of the j-th entry is set to 1 when a legitimate packet with i hop-count gets hashed to j by k hash functions. (Fig. (a))
- ☐ Given a m size table with k hash functions, *NetHCF* can work at a relatively low false positive rate f to keep less than n mappings. (See theoretical analysis in Fig. (b))

(2) The Updating of IP2HC Table

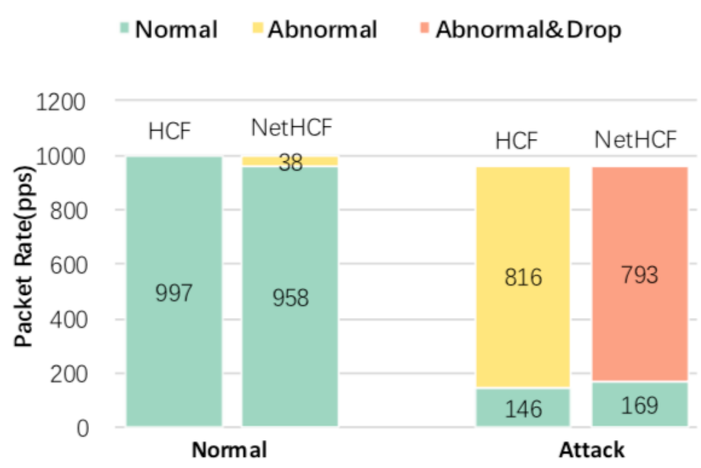
- ☐ *Observation*: considering the access locality, the mappings which have not been accessed in the past period are less likely to be visited in the future. During an update, these mappings should be deleted first.
- ☐ *Basic idea*: Double the bitmap to catch the time-related information and delete the out-of-date mappings timely. (Fig. (c))

(3) Learning and Filtering State

- ☐ In both states, *NetHCF* monitors handshake messages to maintain a TCPSession table. For connections passing three handshakes, NetHCF removes them from the TCPSession table and updates IP2HC to record new mappings.
- ☐ The switching between two states depends on the number of spoofed packets N in a short period T. (if $N > T_{high}$, enter filtering state; if $N < T_{low}$,, re-enter learning state)

❖ *Evaluation and Future work*

NetHCF provides excellent bandwidth protection with low false positive.



Future works:

- ◆ Network-wide coordination
- ◆ TCP connections with asymmetric paths