Optimal Filtering of Source Address Prefixes: Models and Algorithms

Fabio Soldo, Athina Markopoulou
University of California, Irvine
{*fsoldo, athina*}*@uci.edu*

Katerina Argyraki
EPFL, Switzerland
katerina.argyraki@epfl.ch

***Abstract*— How can we protect the network infrastructure from malicious traffic, such as scanning, malicious code propagation, and distributed denial-of-service (DDoS) attacks? One mechanism for blocking malicious traffic is filtering: access control lists (ACLs) can selectively block traffic based on fields of the IP header. Filters (ACLs) are already available in the routers today but are a scarce resource because they are stored in expensive ternary content addressable memory (TCAM). In this paper, we develop, for the first time, a framework for studying filter selection as a resource allocation problem. Within this framework, we study four practical cases of source address/prefix filtering, which correspond to different attack scenarios and operator's policies. We show that filter selection optimization leads to novel variations of the multidimensional knapsack problem and we design optimal, yet computationally efficient, algorithms to solve them. We also evaluate our approach using data from *Dshield.org* and demonstrate that it brings significant benefits in practice. Our set of algorithms is a building block that can be immediately used by operators and manufacturers to block malicious traffic in a cost-efficient way.**

I. Introduction

How can we protect our network infrastructure from malicious traffic, such as scanning, malicious code propagation, spam, and distributed denial-of-service (DDoS) attacks? These activities cause problems on a regular basis ranging from simple annoyance to severe financial, operational and political damage to companies, organizations and critical infrastructure. In recent years, they have increased in volume, sophistication, and automation, largely enabled by botnets that are used as the platform for launching these attacks.

Protecting a victim (host or network) from malicious traffic is a hard problem that requires the coordination of several complementary components, including non-technical (*e.g.*, business and legal) and technical solutions (at the application and/or network level). Filtering support from the network is a fundamental building block in this effort. For example, the victim's internet service provider (ISP) may install filters to react to an ongoing attack, by blocking malicious traffic before it reaches the victim. Another ISP may want to proactively identify and block the malicious traffic before it reaches and compromises vulnerable hosts in the first place. In either case, filtering is a necessary operation that must be performed within the network.

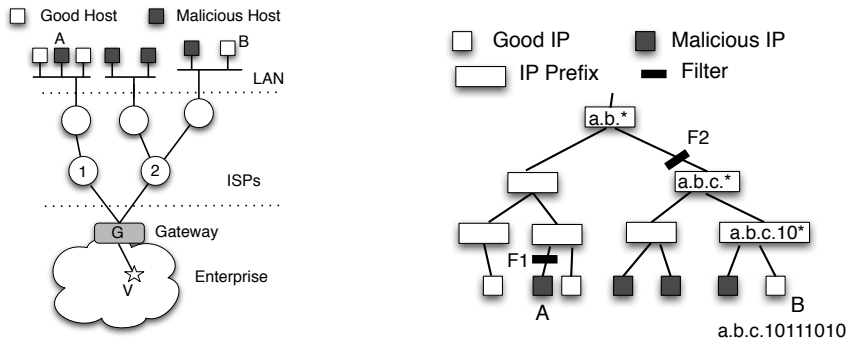
Filtering capabilities are already available at routers today via access control lists (ACLs), which allow a router to match a packet header against rules [1] and are currently used for enforcing a variety of policies, including infrastructure protection [2]. For the purpose of blocking malicious traffic, a filter can be a simple ACL rule that denies access to a source IP address or prefix. To keep up with the high rates of modern routers, it is important that filtering be implemented in hardware: indeed ACLs are stored in the ternary content addressable memory (TCAM), which allows for parallel access and reduces the number of lookups per forwarded packet. However, TCAM is more expensive and consumes more space and power than conventional memory. The size and cost of TCAM puts a limit on the number of filters and this is not expected to change in the near future.[1] With thousands or tens of thousands of filters per path, an ISP alone cannot even hope to block some of the currently witnessed attacks, not to mention attacks from multimillion-node botnets expected in the near future.

Consider the example shown in Fig.1(a): an attacker commands a large number of compromised hosts to send traffic towards a victim V (say a webserver), thus exhausting the resources of V and preventing it from serving its legitimate clients; the ISP of V tries to protect its client from the attack, by blocking the malicious traffic at the gateway router G. Ideally, G would like to assign a single filter to block each malicious IP source. However, there are fewer filters than attackers so aggregation is typically used: a single filter blocks an entire source address prefix. This has the desired effect of reducing the number of filters, but also the undesired side-effect of blocking legitimate traffic originating from that prefix. Therefore, filter selection becomes an optimization problem that tries to block as many malicious and as few legitimate sources as possible, given a certain budget on the number of filters.

In this paper, we formulate, for the first time, a general framework for studying filter selection as a resource allocation problem. To the best of our knowledge, the optimal filter selection aspect has not been explored so far, as most related

This work was supported by the NSF CyberTrust grant 0831530 and by a fellowship from the Networked Systems program at UCI.

[1] A router linecard or supervisor-engine card typically supports a single TCAM chip with tens of thousands of entries. For example, the Cisco Catalyst 4500, a mid-range switch, provides a 64,000-entry TCAM to be shared among all its interfaces (48- 384). Cisco 12000, a high-end router used at the Internet core, provides 20,000 entries that operate at line-speed per linecard (up to 4 Gigabit Ethernet interfaces). The Catalyst 6500 switch can fit 16K-32K patterns and 2K-4K masks in the TCAM. Depending on how an ISP connects to its clients, it can devote to each customer only part of these ACLs, *i.e.*, a few hundreds to a few thousands filters.



(a) Actual network. (b) Hierarchy of source IP addresses and prefixes

Fig. 1. Example of a distributed attack. Let's assume that the gateway router G has only two filters available to block malicious traffic and protect the victim V. It uses $F1$ to block a single malicious address (A) and $F2$ to block prefix $a.b.c.*$, which contains 3 malicious sources but also one legitimate source (B). Therefore, the selection of filter $F2$ trades-off the collateral damage (blocking B) for the reduction in the number of filters (from 3 to 1).

work on filtering has focused on protocol and architectural aspects. Within this framework, we consider four practical source address filtering problems, depending on the attack scenario and the operator's policy and constraints. Our contributions are twofold. On the theoretical side, filter selection optimization leads to novel variations of the multidimensional knapsack problem, and we exploit the special structure of each problem to design optimal and computationally efficient algorithms. On the practical side, we provide a set of cost-efficient algorithms that can be used both by operators to block malicious traffic and by router manufacturers to optimize the use of their TCAM and eventually to optimize the cost of the routers. We would like to emphasize that we do not propose a novel architecture for dealing with malicious traffic; instead, we optimize the use of an important mechanism that already exists on the Internet today and can be immediately used as a building block in larger defense systems.

The structure of the paper is as follows. In Section II, we formulate the general framework for studying filter selection. In Section III, we study four specific problems that correspond to different attack scenarios and operator's policies: blocking all addresses in a blacklist (BLOCK-ALL); blocking some addresses in a blacklist (BLOCK-SOME); blocking flows during a DDoS flooding attack to meet bandwidth constraints (FLOODING); and distributed filtering across several routers during flooding (DIST-FLOODING). For each problem, we design an optimal, yet computationally efficient, algorithm to solve it. In Section IV, we use data from Dshield.org [3] to evaluate the performance of our algorithms in realistic attack scenarios and demonstrate that they bring significant benefit in practice. In Section V, we position our work within (a) the bigger picture of defense against malicious traffic and (b) related knapsack problems. Section VI concludes the paper.

II. Problem Formulation and Framework

A. Definitions and Notation

Let us first define the notation used throughout the paper, also summarized in Table I.

Source IP addresses and prefixes. Every IPv4 address i is a 32-bit sequence. Using the standard notation IP/mask we use p/l to denote a prefix p of length l bits; p and l can take values $l = 0, 1, ..., 32$ and $p = 0, 1, ..., 2^l - 1$ respectively. Sometimes, for brevity, we write simply p to indicate prefix p/l. We write $i \in p/l$ to indicate that address i is within the 2^{32-l} addresses covered by prefix p/l.

Blacklists. A blacklist ($\mathcal{BL}$) is a list of N unique malicious source IP addresses that send malicious traffic towards the victim. Identifying which sources are malicious and should be blocked is a difficult problem on its own right, but orthogonal to the focus of this paper. We consider that the set of malicious IP sources is accurately identified by another module (*e.g.*, an intrusion detection system and/or historical data) in a pre-processing step and is given as input to our problem.

An address is considered "bad" if it appears in a blacklist or "good" if it belongs to a whitelist (set of legitimate addresses), $\mathcal{WL}$. The whitelist may or may not be explicitly given; in the latter case, it includes all addresses that are not in $\mathcal{BL}$.

Address Weight. In the simplest version of the problem, an address is simply either bad or good, depending on whether it appears or not in a blacklist respectively. In a more general framework, a weight w_i can be assigned to every address i to indicate the importance of an address. We use $w_i \leq 0$ for every bad address i to indicate the benefit from blocking it; we use $w_i \geq 0$ for every good address i to indicate the collateral damage from blocking it; $w_i = 0$ indicates indifference about whether address i will be blocked or not.

The weight w_i can have different interpretation depending on the problem. First, it can capture the amount of bad/good traffic originating from an IP address and therefore the benefit/cost of blocking that address. Second, w_i can express policy: *e.g.*, depending on the amount of money gained/lost by the ISP when blocking address i, the operator can decide to assign large positive weights to its important customers that should not be blocked, or large negative weights to the worst attackers that must be blocked [2].

Filters. In this paper, we focus on source address/prefix

[2]The higher the absolute value of the weight assigned to an individual bad/good address, the higher preference to block/not block that address. If all good and bad addresses are assigned the same w_g and $-w_b$ respectively, then the ratio $\frac{w_g}{w_b}$ is a parameter that the operator can tune to express how much she values low collateral damage vs. blocked malicious traffic. $w_i = \infty$ ($-\infty$) indicates that address i must never (always) be blocked.

i	Generic IP address
w_i	Weight assigned to address i
$\mathcal{BL}$	Blacklist: a list of "bad" addresses
N	Number of unique addresses in $\mathcal{BL}$
$\mathcal{WL}$	Whitelist: a list of "good" addresses
p/l (or "p" for short)	prefix p of length l bits (IP/mask notation)
$i \in p/l$	address i that belongs to prefix p/l
$x_{p/l} \in \{1, 0\}$	indicates if a filter blocks prefix p/l or not
$g_{p/l} = \sum_{i \in p/l \cap \mathcal{WL}} w_i$	collateral damage from filtering prefix p/l
$b_{p/l} = \lvert \sum_{i \in p/l \cap \mathcal{BL}} w_i \rvert$	bad traffic blocked by filtering prefix p/l
F_{max}	Maximum number of available filters
$z_p(F)$ (or $z_p(F, C)$)	optimal solution of subproblem considering only addresses in prefix p and F filters (and capacity C, in the case of FLOODING)

TABLE I
NOTATION

filtering. A filter is a simple ACL rule that specifies that all addresses in prefix p/l should be blocked. F_{max} denotes the maximum number of filters available in TCAM for our purpose, and is given as input to our problem. Notice that filter optimization is only meaningful when the number of available filters F_{max} is much smaller than the number of malicious sources N, which is indeed the case in practice (see introduction and [1], [2]).

The decision variable $x_{p/l} \in \{0, 1\}$ is 1 if a filter is assigned to block prefix p/l; or 0 otherwise. A filter p/l blocks all 2^{32-l} addresses in that range. This has the desired effect of blocking all bad traffic $b_{p/l} = \lvert \sum_{i \in p/l \cap \mathcal{BL}} w_i \rvert$ and the side-effect of blocking all legitimate traffic $g_{p/l} = \sum_{i \in p/l \cap \mathcal{WL}} w_i$, originating from that prefix.

B. Rationale and Overview of Filtering Problems

Given a set of malicious and legitimate sources, and a measure of their importance (w's), the goal of filter selection is the construction of filtering rules, so as to minimize the impact of malicious sources on the network using the available network resources (*e.g.*, filters and link capacity). Depending on the attack scenario, and the operator's policy and constraints, different problems may arise. *E.g.*, the operator may want to block all malicious sources, or may tolerate to leave some unblocked; the attack may be of a low rate or a flooding attack; the operator may control one or several routers.

In the core of each filtering problem lies the following optimization problem:

$$\min \sum_{p/l} \sum_{i \in p/l} w_i \cdot x_{p/l} \tag{1}$$

$$\text{s.t.} \sum_{p/l} x_{p/l} \leq F_{max} \tag{2}$$

$$\sum_{p/l : i \in p/l} x_{p/l} \leq 1 \quad \forall i \in \mathcal{BL} \tag{3}$$

$$x_{p/l} \in \{0, 1\} \quad \forall l = 0, ..32, p = 0, ..2^l \tag{4}$$

Eq.(1) expresses the objective to minimize the total cost for the network, which consists of two parts: the collateral damage (terms with $w_i > 0$) and the cost of leaving malicious traffic unblocked (terms with $w_i < 0$). We denote summation over all possible prefixes p/l: $l = 0, ...32$, $p = 0, ...2^l - 1$ by, $\sum_{p/l}$. Eq.(2) expresses the constraint on the number of filters. Eq.(3) states that overlapping filters are mutually exclusive, *i.e.*, each malicious address should be blocked at most once, otherwise filtering resources are wasted. Eq.(4) lists the decision variables $x_{p/l}$ corresponding to all possible prefixes; it is part of every optimization problem in this paper and will be omitted from now on for brevity.

Eq.(1)-(4) provide the general framework for filter selection optimization. Different filtering problems can be written as special cases within this framework, possibly with additional constraints. As we discuss in Section V-B, these are all multi-dimensional knapsack problems [4], which are in general, NP-hard. The specifics of each problem affect dramatically the complexity, which can vary from linear to NP-hard.

In this paper, we formulate four practical filtering problems, and we develop optimal, yet computationally efficient algorithms to solve them. Here, we summarize the rationale behind each problem and our main results. The exact formulation and detailed solution for each problem is provided in section III.

[P_1] *BLOCK-ALL*: Assume that a blacklist $\mathcal{BL}$ and a whitelist $\mathcal{WL}$ are given; a weight is also associated with every good address to indicate the amount of legitimate traffic originating from that address. The limit on the number of filters is F_{max}. The operator wants to choose a set of filters that block *all* malicious sources so as to minimize the collateral damage. We design an optimal algorithm that solves this problem at low-complexity (linearly increasing with N, *i.e.*, the lowest asymptotical complexity for this problem).

[P_2] *BLOCK-SOME*: Assume that the same blacklist and whitelist are given, as in P_1. However, the operator may be willing to block *only some* (instead of all) malicious addresses, so as to decrease the collateral damage, at the expense of leaving some malicious traffic unblocked. She can achieve this by assigning weights $w_i > 0$ and $w_i < 0$ to good and bad addresses, respectively, to express their relative importance. The goal of P_2 is to block only those subsets of malicious addresses that have the highest impact and are not co-located with important legitimate sources, so as to minimize the total cost in Eq.(1). We design an optimal, computationally efficient (linearly increasing with N) algorithm for this problem too.

[P_3] *FLOODING*: In a distributed *flooding attack*, such as the one shown in Fig.1, a large number of compromised hosts send traffic to the victim with the purpose of exhausting the victim's access bandwidth. The problem is well-known and increasingly frequent and severe. Our framework can be used to optimally select filters in this case, so as to minimize the collateral damage and meet the bandwidth constraint (*i.e.*, the total bandwidth of the unblocked traffic should not exceed the bandwidth of the flooded link, *e.g.*, link G-V in Fig.1). The input is the same as in P_1-P_2, and the weights capture the *traffic volume* originating from each IP source. We prove that the problem P_3 is NP-hard and we design a pseudo-polynomial algorithm that optimally solves FLOODING, and whose complexity grows linearly with the number of traffic sources.

[P_4] *DIST-FLOODING*: All the above problems aim at

selecting filters at a single router. However, a network administrator, of an ISP or campus network, may use the filtering resources collaboratively across *several routers* to better defend against an attack. The question then is not only which filters to select but also on which router to place them. Here, we focus distributed filtering, across several routers, against a flooding attack. We prove that P_4 can be decomposed into several FLOODING problems, that can be solved independently and optimally, one at each router.

III. Filtering Problems and Algorithms

In this section, we give the detailed formulation of each problem and the algorithm that solves it. But first, let us define a data structure that we use to represent the problem and to develop all the subsequent algorithms.

A. Data Structure for Representing the Problem

Definition 1 (LCP Tree): Given a set $\mathcal{A}$ of N IP addresses, we define the *Longest Common Prefix tree* of $\mathcal{A}$, LCP($\mathcal{A}$), as the binary tree whose leaves represent the N IPs and all other nodes represent all and only the longest common prefixes between any pair of IPs in $\mathcal{A}$. The prefixes are organized in a hierarchy, with shorter prefixes towards the root and longer prefixes towards the leaves, so that the prefix corresponding to a parent node includes the prefixes corresponding to its two children.

An example is shown and discussed in Fig.2. The LCP tree can be constructed from the binary tree of all prefixes, by removing the branches that do not have malicious IPs and then by removing nodes with a single child. It reduces the storage for representing candidate prefixes by encoding those prefixes that are part of a feasible solution [5] . We do not claim novelty in this data structure but we describe it in detail because we use it extensively in the design of the algorithms.

Complexity: We can build the LCP tree from N malicious addresses by performing N insertions in a Patricia trie [5]. To insert a string of m bits, we need at most m comparisons. Thus, the worst case complexity is $O(mN)$, where $m = 32$ (bits) is the constant length of an IP address.

We will make extensive use of the LCP tree in all algorithms in the rest of this section, as it provides a compact way to represent feasible solutions and to efficiently select the optimal one. Note that every node in the LCP-tree is a candidate prefix p/l; for brevity of notation, we will use interchangeably the notation p/l and its shorter version p.

B. BLOCK-ALL

Goal. Given: (i) a blacklist of malicious addresses $\mathcal{BL}$ (ii) a set of legitimate sources (iii) weights assigned to each legitimate source address indicating the amount of traffic from that address and (iv) a limit on the number of filters F_{max}; select source address prefixes so as to block *all* malicious sources and minimize the collateral damage.

Formulation. This can be formulated within the general framework of Eq.(1)-(4) by assigning $w_i > 0$ to good addresses (the amount of legitimate traffic) and weight $w_i = 0$

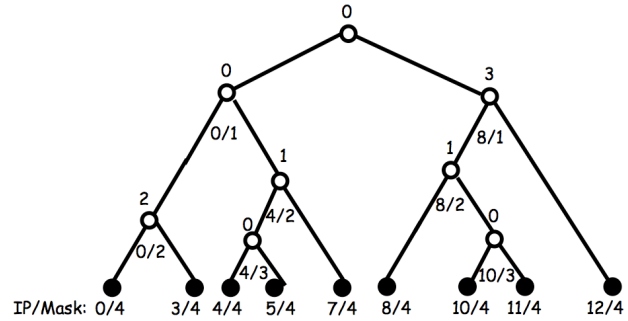


Fig. 2. **Example of LCP-tree used in BLOCK-ALL.** For ease of illustration, consider a 4-bit (instead of 32-bit) address space, i.e. from 0000 to 1111. Let $\mathcal{BL} = \{0, 3, 4, 5, 7, 810, 11, 12\}$ be the set of malicious IPs, corresponding to the leaves of the binary tree. All remaining IPs $(1, 2, 6, 9, 13, 14, 15)$ are considered legitimate and not explicitly shown. Every intermediate node represents the longest common prefix (LCP) covering all malicious sources in that subtree; it is associated with a cost measuring the additional collateral damage caused when we filter that node, instead of filtering each of its children. E.g. the LCP of malicious addresses 0=0000 and 3=0011 is prefix 00**; if filter 00** is chosen instead of filters 0000 and 0011, collateral damage of 2 is caused, because the legitimate addresses 1 and 2 are also blocked. Choosing a set of source prefixes to filter is equivalent to choosing a set of nodes in this LCP tree.

to each malicious source. The goal is to minimize the total cost, which in this case is simply the total legitimate traffic blocked: $\sum_{p/l} \sum_{i \in p/l} w_i \cdot x_{p/l} = \sum_{i \in p/l \cap \mathcal{WL}} w_i + 0 = g_{p/l}$. Constraint Eq.(7) enforces that every malicious source should be blocked by exactly one filter.

$$\min \sum_{p/l} g_{p/l} x_{p/l} \tag{5}$$

$$\text{s.t.} \quad \sum_{p/l} x_{p/l} \leq F_{max} \tag{6}$$

$$\sum_{p/l : i \in p/l} x_{p/l} = 1 \qquad \forall i \in \mathcal{BL} \tag{7}$$

Characterizing an Optimal Solution. We search for solutions that can be represented as a subtree of the LCP tree structure, as described in the following:

Proposition 3.1: Given $\mathcal{BL}$ and F_{max}, there exists an optimal solution of BLOCK-ALL that can be represented as a pruned subtree of LCP($\mathcal{BL}$) with: the same root, up to F_{max} leaves, and non-leaf nodes having exactly two children.

Proof Sketch. The main idea of the proof is that every feasible solution of BLOCK-ALL can be reduced to another feasible solution that (i) corresponds to a subtree of LCP($\mathcal{BL}$) as described in Prop. 3.1, and (ii) has smaller or equal collateral damage. Clearly, every feasible solution, S, of Eq. (5)-(7) can be represented as a pruned subtree of the binary tree of all possible IP prefixes, with the same root and leaves being the prefixes used as filters. Assume that S uses a prefix $\tilde{p}/\tilde{l}$ which is not in LCP($\mathcal{BL}$). Therefore, either $\tilde{p}/\tilde{l}$ does not contain any bad IPs or one of its two branches does not. In the former case, we can remove filter $\tilde{p}/\tilde{l}$, as it is not blocking any bad IPs. In the latter case, we can move the filter from $\tilde{p}/\tilde{l}$ to the child that contains bad IPs. In both cases, we have a new

feasible solution, with smaller or equal collateral damage than the original solution. Iterating this process until all prefixes are in the LCP shows that any feasible solution can be transformed to a feasible solution corresponding to a subtree of LCP($\mathcal{BL}$), as described in the proposition and having smaller or equal collateral damage. Therefore, an optimal solution, which is obviously also a feasible one, can be transformed to that form. Finally, we note that in every subtree corresponding to a feasible solution, each node must have two or zero children. The reason is that a node with only one child, would leave a branch unfiltered in the LCP tree, which would leave at least one bad IP unfiltered. Further details are provided in [6]. ■

Algorithm. Algorithm 1, which solves BLOCK-ALL, consists of two main steps. First, we build the LCP-tree from the input blacklist. Second, in a bottom-up fashion, we compute $z_p(F) \forall p, F$, i.e. the minimum collateral damage needed to block all malicious IPs in the subtree of prefix p using at most F filters. Following a dynamic programming (DP) approach, we can find the optimal allocation of filters in the subtree rooted at prefix p, by finding a value n and by assigning $F-n$ filters to the left subtree and n to the right subtree, so as to minimize the collateral damage. The fact that we need to filter all malicious addresses (leaves in the LCP tree) implies that at least one filter must be assigned to the left and right subtree, i.e. $n = 1, 2..., F-1$.

For every pair of sibling nodes, s_l (left) and s_r (right), with common parent node p, we have the DP recursive equation:

$$z_p(F) = \min_{n=1,\dots,F-1} \Big\{ z_{s_l}(F-n) + z_{s_r}(n) \Big\}, \; F > 1 \quad (8)$$

with boundary conditions for leaf and intermediate nodes:

$$z_{leaf}(F) = 0 \; \forall F \geq 1, \quad z_p(1) = g_p \; \forall p \quad (9)$$

Once we compute $z_p(F)$ for all prefixes in the LCP-tree, we simply read the value of the optimal solution, $z_{root}(F_{max})$. We also use the variables $X_p(F)$ to keep track of the set of prefixes used in the optimal solution. In lines (4) and (10) of Algorithm 1, $X_p(F)$ is initialized to the single prefix used. In line (12), after computing the new cost, the corresponding set of prefixes is updated: $X_p(F) = X_{s_l}(F-n) \cup X_{s_r}(n)$.

Algorithm 1 computes the optimal solution of problem BLOCK-ALL: the prefixes contained in set $X_p(F)$ are the optimal $x_{p/l} = 1$ for Eq.(5)-(7). The proof is straightforward based on the DP Eq.(8) and omitted for lack of space [6].

Complexity. Computing Eq.(8) for every node p and for every $F \in [1, F_{max}-1]$ involves solving $N(F_{max}-1)$ subproblems, one for every pair (p, F), with complexity $F_{max}-1$ each. Proceeding from the leaves to the root, we can compute the optimal solution in $N(F_{max}-1)^2$. This bound can be made tighter observing that we only need to compute $z_p(F)$ for $F \leq \min\{|leaves(p)|, F_{max}\}$, where $|leaves(p)|$ is the number of the leaves under prefix p in the LCP-tree. Using this observation, it can be shown that the computation can be done in $O(NF_{max})$, which is essentially $O(N)$, since $F_{max} << N$ and F_{max} does not depend on N but only on the TCAM size. Thus, the time complexity increases linearly with the number of malicious IPs N. Within a constant factor, this is the lowest achievable complexity, since we need to read all N malicious IPs at least once [6].

Algorithm 1 *Algorithm for BLOCK-ALL*

```
1:  build LCP-tree(BL)
2:  for all leaf nodes leaf do
3:      z_leaf(F) = 0 ∀F ∈ [1, F_max]
4:      X_leaf(F) = {leaf} ∀F ∈ [1, F_max]
5:  end for
6:  level = level(leaf)-1
7:  while level ≥ level(root) do
8:      for all node p such that level(p)==level do
9:          z_p(1) = g_p
10:         X_p(1) = {p}
11:         z_p(F) = min_{n=1,..F-1} { z_{s_l}(F − n) + z_{s_r}(n) }   ∀F ∈
            [2, F_max]
12:         X_p(F) = X_{s_l}(F − n) ∪ X_{s_r}(n)  ∀F ∈ [2, F_max]
13:     end for
14:     level = level - 1
15: end while
16: return z_root(F_max), X_root(F_max)
```

C. BLOCK-SOME

Goal. Given: (i) a blacklist of malicious addresses (ii) a set of legitimate sources (iii) weights assigned to all addresses, which express relative importance and (iv) a limit on the number of filters F_{max}; block some source address prefixes so as to minimize the total cost, including the collateral damage and the benefit of blocking malicious addresses.

Formulation. This can be formulated within the general framework of Eq.(1)-(4), by assigning to good and bad addresses weights $w_i > 0$ and $w_i < 0$ respectively, to express their relative importance. The goal is to minimize the total cost, as in Eq.(1), which in this case includes both collateral damage $g_{p/l}$ and unfiltered malicious traffic $b_{p/l}$.

$$\min \sum_{p/l} \Big(g_{p/l} - b_{p/l} \Big) x_{p/l} \quad (10)$$

$$\text{s.t.} \sum_{p/l} x_{p/l} \leq F_{max} \quad (11)$$

$$\sum_{p/l : i \in p/l} x_{p/l} \leq 1 \qquad \forall i \in \mathcal{BL} \quad (12)$$

Another difference from BLOCK-ALL is Eq.(12), which dictates that every malicious source must be covered at most by one prefix, but does not necessarily have to be covered.

Characterizing an Optimal Solution. We can leverage again the structure of the LCP tree to characterize feasible and optimal solutions, with a proposition similar to Prop. 3.1. The only difference from BLOCK-ALL is that, because some bad IPs can remain unfiltered, the pruned subtree corresponding to a feasible solution can now have nodes with a single descendant. We defer details to [6].

Algorithm. The algorithm is similar to Algorithm 1 in that it uses the LCP-tree and a similar DP approach. The difference is that not all addresses need to be covered and, at each step, we can assign $n = 0$ filters to the left or right subtree, i.e. in

line (11) of Algorithm 1: $n = 0, 1..., F$. We can recursively compute the optimal solution as before:

$$z_p(F) = \min_{n=0,\ldots,F} \Big\{ z_{s_l}(F-n) + z_{s_r}(n) \Big\} \quad (13)$$

with boundary conditions for intermediate (p) and leaf nodes:

$$z_p(0) = 0 \quad \forall\ p, \qquad z_{leaf}(F) = -b_{leaf} \quad \forall F \geq 1 \quad (14)$$

$$z_p(1) = \min \Big\{ g_p - b_p, \min_{n=0,1} \Big\{ z_{s_l}(1-n) + z_{s_r}(n) \Big\} \Big\} \quad (15)$$

Complexity. The analysis of BLOCK-ALL can be applied to this algorithm as well. The complexity turns out to be the same, i.e. linearly increasing in N as well [6].

BLOCK-ALL vs. BLOCK-SOME. There is an interesting connection between the two problems. The latter can be regarded as an automatic way to select the best subset from $\mathcal{BL}$ in terms of the weights w_i, and then run BLOCK-ALL only on that subset. The advantage is that we do not need to search for the optimal subset, which is automatically given in the final solution. In the extreme case that much more importance is given to the bad rather than the good addresses, BLOCK-SOME degenerates to BLOCK-ALL.

D. FLOODING

Goal. Given: (i) a blacklist of malicious addresses (ii) a set of legitimate sources (iii) *the amount of traffic* that each generates (iv) a limit on the number of filters F_{max} and (v) a constraint on the link capacity (bandwidth) C; block some source address prefixes so as to minimize the collateral damage and make the total traffic fit within the link capacity.

Formulation.

$$\min \sum_{p/l} g_{p/l} x_{p/l} \quad (16)$$

$$\text{s.t.} \sum_{p/l} x_{p/l} \leq F_{max} \quad (17)$$

$$\sum_{p/l} \Big(g_{p/l} + b_{p/l} \Big)(1 - x_{p/l}) \leq C \quad (18)$$

$$\sum_{p/l : i \in p/l} x_{p/l} \leq 1 \quad \forall i \in \mathcal{BL} \quad (19)$$

where, $g_{p/l}$ and $b_{p/l}$ denote the amount of good bad traffic from prefix p/l, respectively. Eq.(19) indicates that we are interested in blocking some, not all, malicious sources, and that we should not use overlapping prefixes. Before the attack, the total good traffic $t_0 = \sum_{p/l} \Big(g_{p/l} + b_{p/l} \Big)$ could fit within the capacity; after flooding, the total traffic exceeds the capacity. Eq.(18) says that the total traffic that remains unblocked after filtering should fit within the link capacity C.

Characterizing an Optimal Solution. We use the LCP tree for all addresses $\mathcal{BL} \cup \mathcal{WL}$. Furthermore, to account for Eq.(18), we assign a cost, t_p, to every node in the LCP tree, representing the total traffic, $t_p = g_p + b_p$, generated by prefix p/l.

Proposition 3.2: Given $\mathcal{BL}$, $\mathcal{WL}$, F_{max}, and C, there exists an optimal solution of the FLOODING problem that can be represented as a pruned subtree of LCP($\mathcal{BL} \cup \mathcal{WL}$), with the same root, up to F_{max} leaves, and such that the total cost of the leaves be $\geq t_0 - C$.

Proof. The proof is along the same lines as Prop.3.1 [6]. ■

Algorithm. FLOODING is a 2-dimensional knapsack problem (2KP), with an additional capacity constraint, Eq.(19), that makes it harder. 2KP is a "very hard" problem: not only it is NP-Hard, but also the existence of a full polynomial time approximation scheme for this problem is unlikely to exist, since it would imply that $\mathcal{P} = \mathcal{NP}$ [7]. For FLOODING we obtain the following hardness result:

Theorem 3.3: The optimization problem FLOODING in Eq.(16)-(19) is NP-Hard.

Proof: It is obvious that FLOODING is in $\mathcal{NP}$. To prove that it is also $\mathcal{NP}$-hard, we consider the KP problem with a cardinality constraint:

$$\max \sum_{i \in I} p_i x_i, \quad \text{s.t.} \sum_{i \in I} w_i x_i \leq C_1 \text{ and } \sum_{i \in I} x_i = k \quad (20)$$

which is known to be $\mathcal{NP}$-hard [4], and we show that it reduces to FLOODING. First, note that any solution of the above problem that uses $F < F_{max}$ filters can be transformed to another feasible solution with exactly F_{max} filters, without increasing the collateral damage. Therefore, the inequality in Eq.(17) can be replaced by an equality without affecting the collateral damage of the optimal solution. Second, we define $\bar{x}_{p/l} = 1 - x_{p/l}$, $\bar{F}_{max} = \Big(\sum_{p/l} 1 \Big) - F_{max}$ and we rewrite the above problem:

$$\max \sum_{p/l} g_{p/l} \bar{x}_{p/l} \text{ s.t. : } \sum_{p/l} \bar{x}_{p/l} = \bar{F}_{max},$$

$$\sum_{p/l} \Big(g_{p/l} + b_{p/l} \Big) \bar{x}_{p/l} \leq C, \quad \sum_{p/l : i \in p/l} \bar{x}_{p/l} \leq 1 \ \forall i \in \mathcal{BL}$$

For a given instance of Problem (20), we construct an equivalent instance of the above problem by introducing the following mapping. For $i = 1, \ldots, N$: $-\bar{g}_{ii} = p_i$, $(g_{ii} + b_{ii}) = w_i$. For p/l that is not in the blacklist: $\bar{g}_{p/l} = 0$ and $(g_{p/l} + b_{p/l}) = C + 1$. Moreover, we assign $\bar{F}_{max} = k$ and $C = C_1$. With this assignment a solution to the KP problem (20) can be obtained by solving FLOODING and then taking the values of variables $x_{p/l}$ such that p/l is in the blacklist. ■

Therefore, we do not to look for a polynomial time algorithm. Instead, we designed the following dynamic programming algorithm that optimally solves FLOODING.

Let $z_p(F, c)$ be the minimum collateral damage solving FLOODING problem with F filters and capacity c:

$$z_p(F, c) = \min_{\substack{n=0,\ldots,F \\ m=0,\ldots,c}} \{ z_{s_l}(F-n, c-m) + z_{s_r}(n, m) \} \quad (21)$$

Complexity. The DP approach computes $O(CF_{max})$ entries for every node. Moreover, the computation of a single entry, given the entries of descendant nodes, require $O(CF_{max})$ operations, Eq.(21). Therefore, the optimal solution, $z_{root}(F_{max}, C)$, can be computed in $O((N + |\mathcal{WL}|)C^2)$

time. The algorithm has pseudo-polynomial complexity since it is polynomial in C that cannot be bounded by the input length. More importantly, its complexity increases linearly with the number of IP sources in $\mathcal{BL} \cup \mathcal{WL}$.

FLOODING vs. BLOCK-SOME. To see the connection between FLOODING and BLOCK-SOME, let us consider a partial Lagrangian relaxation of (16)-(19):

$$\max_{\lambda \geq 0} \Big\{ \min \sum_{p/l} \Big[(1-\lambda) g_{p/l} - \lambda b_{p/l} \Big] x_{p/l} + \\ + \sum_{p/l} \lambda (g_{p/l} + \lambda b_{p/l}) - \lambda C \Big\} \quad (22)$$

$$\text{s.t.} \sum_{p/l} x_{p/l} \leq F_{max} \quad (23)$$

$$\sum_{p/l : i \in p/l} x_{p/l} \leq 1 \quad \forall i \in \mathcal{BL} \quad (24)$$

For every fixed $\lambda \geq 0$ problem (22)-(24) is equivalent to (16)-(19) for a specific assignments of weights w_i. This shows that dual feasible solutions of FLOODING are instances of BLOCK-SOME for a particular assignment of weights. The dual problem, in the variable λ, aims exactly at tuning the Lagrangian multiplier to find the best assignment of weights.

E. DISTRIBUTED-FLOODING

Goal: Consider a victim V that connects to the Internet through its ISP and is flooded by a set of attackers listed in a blacklist $\mathcal{BL}$, as in Fig.1(a). To reach the victim, attack traffic has to pass through one or more ISP routers; let $\mathcal{R}$ be the set of unique routers from some attacker to the victim. Let each router $u \in \mathcal{R}$ have capacity $C^{(u)}$ on the downstream link (towards V) and a limited number of filters $F^{(u)}_{max}$. We assume that the volume of good/bad traffic through every router is known. Our goal is to allocate filters across all routers, in a distributed way, so as to minimize the total collateral damage and avoid congestion on all links of the ISP network.

Formulation. Let the variables $x^{(u)}_{p/l} \in \{0,1\}$ indicate whether or not filter p/l is used at router u. Then the distributed filtering problem can be stated as:

$$\min \sum_{u \in \mathcal{R}} \sum_{p/l} g^{(u)}_{p/l} x^{(u)}_{p/l} \quad (25)$$

$$\text{s.t.} \sum_{p/l} x^{(u)}_{p/l} \leq F^{(u)}_{max} \qquad \forall u \in \mathcal{R} \quad (26)$$

$$\sum_{p/l} \Big(g^{(u)}_{p/l} + b^{(u)}_{p/l} \Big) (1 - x^{(u)}_{p/l}) \leq C^{(u)} \qquad \forall u \in \mathcal{R} \quad (27)$$

$$\sum_{u \in \mathcal{R}} \sum_{p/l} x^{(u)}_{p/l} \leq 1 \qquad \forall i \in \mathcal{BL} \quad (28)$$

Characterizing an Optimal Solution. Given the sets $\mathcal{BL}$, $\mathcal{WL}$, $\mathcal{R}$, and $F^{(u)}_{max}$, $C^{(u)}$ at each router, we have the following:

Proposition 3.4: There exists an optimal solution of DIST-FLOODING that can be represented as a set of $|\mathcal{R}|$ different pruned subtrees of the LCP-tree($\mathcal{BL} \cup \mathcal{WL}$), each corresponding to a feasible solution of FLOODING for the same input, and s.t. every subtree leaf is not a node of another subtree.

Proof. See [6]. ■

Algorithm. Constraint (28), which imposes that different routers do not block the same prefixes, prevents us from a direct decomposition of the problem. To decouple the problem, consider the following partial Lagrangian relaxation:

$$L(x, \lambda) = \sum_{u \in \mathcal{R}} \Big(\sum_{p/l} \Big(g^{(u)}_{p/l} + \lambda_{p/l} \Big) x^{(u)}_{p/l} \Big) - \sum_{i \in \mathcal{BL}} \lambda_i$$

where λ_i is the Lagrangian multiplier (price) for the constraint in Eq.(28), and $\lambda_{p/l} = \sum_{i \in p/l} \lambda_i$ is the price associated with prefix p/l. With this relaxation, both the objective function and the other constraints immediately decompose in $|\mathcal{R}|$ independent sub-problems, one per router u:

$$\min \sum_{p/l} \Big(g^{(u)}_{p/l} + \lambda_{p/l} \Big) x^{(u)}_{p/l} \quad (29)$$

$$\text{s.t.} \sum_{p/l} x^{(u)}_{p/l} \leq F^{(u)}_{max} \quad (30)$$

$$\sum_{p/l} \Big(g^{(u)}_{p/l} + b^{(u)}_{p/l} \Big) (1 - x^{(u)}_{p/l}) \leq C^{(u)} \quad (31)$$

The dual problem is:

$$\max_{\lambda_i \geq 0} \sum_{u \in \mathcal{R}} h_u(\lambda) - \sum_{i \in \mathcal{BL}} \lambda_i \quad (32)$$

where $h_u(\lambda)$ is the optimal solution of (29)-(31) for a given λ. Given the prices λ_i, every sub-problem (29)-(31) can be solved independently and optimally by router u using e.g. Eq. (21). The dual problem can be solved using a standard projected subgradient method, as discussed in [4]. Note, however, that since $x \in \{0,1\}$ the dual problem is not always guaranteed to converge to a primal feasible solution [8].

Distributed vs. Centralized Solution. The above formulation lends itself naturally to a distributed implementation. Each router needs to only solve their own subproblem (29)-(31) independently from the others. A single machine (e.g. the victim's gateway or a dedicated node) should solve the master problem (32) to iteratively find the prices that coordinate all subproblems. Thus, at every iteration of the subgradient, the new λ_i's need to be broadcasted to all routers. Given the λ_i's, the routes independently solve a sub-problem each and return the computed $x^{(u)}_{p/l}$ to the node in charge of the master problem. Even in a centralized setting, our distributed scheme is efficient because it lends itself to parallel computation of Eq.(25)-(28).

IV. PRACTICAL EVALUATION

In this section, we use real blacklists to demonstrate that filter optimization brings significant gain. The reason is that, in practice, malicious sources appear clustered in the IP address space, which means that a small number of filters is sufficient to block most malicious IPs with low collateral damage, a feature exploited by our algorithms [9]. Due to lack of space, we only present limited simulation results; however, these results demonstrate the above point, as well as some of the structural properties of the solutions for BLOCK-ALL and BLOCK-SOME, which are at the heart of our framework.

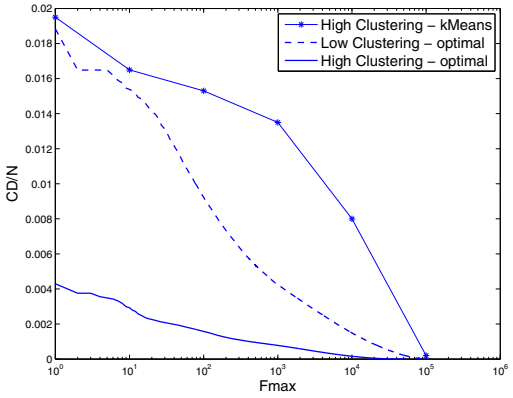


Fig. 3. BLOCK-ALL: collateral damage (CD) normalized over the number of malicious sources (N) vs. number of filters F_{max}. We compare Algorithm 1 to K-means. In particular, we simulated Lloyd's heuristic for K-means, which is NP-hard; we ran 50 runs to avoid local minima. We also run Algorithm 1 on two traces, those with the highest and lowest degree of clustering.

A. Simulation Setup

We analyzed a 61-day trace from *Dshield.org* - a repository of firewall and intrusion detection logs from about 1,700 different organizations. The dataset includes 758,698,491 attack reports, from 32,950,391 different IP sources. Each report includes, among other things, the malicious source IP and the victim's destination IP. We looked at each victim (IP destination) in the dataset; the set of sources attacking each victim serves as a blacklist for our simulations and varies considerably among victims. We also generated good traffic arriving at a domain hosting 20 servers, each receiving an average rate of 1,000 incoming good connections per second, each connection generating 5KB of traffic; we selected the good IP addresses according to the multifractal distribution in [10].

B. Simulation Results

BLOCK-ALL. In Fig. 3, we chose two victims, each attacked by a large number (up to 100,000) of malicious IPs in a single day; we picked these particular two, because the corresponding blacklists exhibit the highest and the lowest degree of attack source clustering observed in the entire dataset. We ran Algorithm 1 on the two blacklists and made the following observations, Fig. 3: First, the optimal algorithm performs significantly better than a generic clustering algorithm that does not exploit the structure of IP prefixes. In particular, it reduces the collateral damage (CD) by up to 85% compared to K-means, when run on the same blacklist. Second, the degree of clustering in a blacklist matters: the CD is lowest (highest) in the blacklist with highest (lowest) degree of clustering, respectively. Results obtained for other victim destinations and days were similar and lied in between these two extremes. A few thousand filters were sufficient to significantly reduce collateral damage (CD) in all cases.

BLOCK-SOME. In Fig. 4, we focus on the blacklist with the least clustering and thus the highest CD (dashed line in Fig. 3). In this worst-case scenario, an alternative to BLOCK-ALL is BLOCK-SOME, which allows the operator to trade-off lower

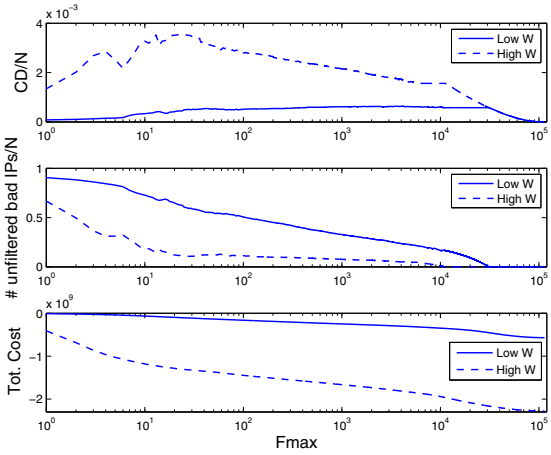


Fig. 4. BLOCK-SOME. (a) Collateral damage (CD) (b) number of unblocked bad IPs (UBIP) (c) total cost ($CD - W \cdot UBIP$). The operator expresses relative tolerance to UBIP vs. CD by tuning the weight $W = \frac{w_b}{w_a}$. We considered a higher (2^{14}) and a lower (2^{10}) value of W.

CD for unblocked bad IPs (UBIP) by appropriately tuning the weights. We ran the BLOCK-SOME algorithm on this blacklist, assigning, for simplicity, the same weights w_g and w_b to every good and bad source. Fig. 4 shows the results: In Fig. 4(a), the CD is always smaller than the corresponding CD in Fig. 3; they become equal only when we block all bad IPs. In Fig. 4(b), we see that BLOCK-SOME reduces the CD by 60% compared to BLOCK-ALL, while leaving unblocked only 10% of bad IPs and using only a few hundred filters. In the data we analyzed, we observed that CD tends to first increase, then decrease with F_{max}, while UBIP tends to decrease.[3] The ratio w_b/w_g captures the effort made by BLOCK-SOME to block all bad IPs and become similar to BLOCK-ALL.

Simulations for FLOODING and DIST-FILTERING will be included in the extended version. However, as already mentioned, these problems are also based on BLOCK-SOME and can therefore leverage the clustering inherent in blacklists.

V. Our Work in Perspective

A. The Bigger Picture of Defense against Malicious Traffic

Dealing with malicious traffic is a hard problem that requires the cooperation of several components. In this paper, we did not propose a novel solution; instead, we optimized the use of filtering - a mechanism that already exists in the Internet today and is a necessary building block of any bigger solution. We focused on the optimal selection of filtering rules, which can then be installed and propagated by filtering protocols [11], [12]. We rely on a detection module to distinguish good from bad traffic and provide us with a blacklist. Detection is a difficult but orthogonal problem to the contribution of this paper.

[3]We can explain this as follows. When a new filter is available, the new optimal solution can be constructed by (i) blocking a new cluster of bad IPs (ii) splitting a blocked cluster into two filters or (iii) a combination of (i)&(ii)& merging of existing filters. For small F_{max}, option (i) is dominant: the inherent clustering allows to find a cluster that is not blocked yet; this increases CD and reduces UBIP. When this is not possible, option (ii) becomes dominant, CD decreases and UBIP remains constant or decreases slowly.

We also consider addresses in the blacklist to be true and not spoofed. This is a reasonable assumption today, because attackers have the luxury to use botnets that involve large numbers of infected hosts, so that they do not need to use spoofing In 2005, less than 20% of addresses were spoofable [13], while in 2008, only 7% of addresses in Dshield logs were found likely to have been spoofed [9]. Looking into the future, there is also a number of proposals for enforcing source accountability, including ingress filtering [14], self-certifying addresses [15], and packet passports [16].

A practical deployment scenario is that of a single network under the same administrative authority, such as an ISP or campus network. The operator can use our algorithms to create filtering rules, at a single or at several routers, in order to optimize the use of its own resources and defend against an attack in a cost-efficient way. Our distributed algorithm may also prove useful, not only for a distributed protocol of routers within the same ISP, but also in the future, when different ISPs start cooperating against common enemies.

The following papers are related to our work. In [17], source filtering via ACLs was studied against DDoS attacks; however, the filters were heuristically selected and the approach was entirely simulation-based. There is a body of work on firewall rule configuration [18], which focuses on management and misconfigurations, not on resource allocation. Furthermore, they consider firewalls for enterprises, which are not supposed to be accessed from outside and thus can be protected without filtering rules. In an earlier workshop paper [19], we also studied optimal source-based filtering by aggregating source addresses into continuous ranges, *not prefixes*. This allowed for greedy solutions but does not reflect the ACL specifications.

B. Relation to Knapsack Problems

Optimal filter selection belongs to the family of multidimensional knapsack problems (dKP) [4]. The general dKP problem is well known to be NP-hard. The most relevant variation to us is the knapsack with cardinality constraint (1.5KP) [20], which has $d = 2$ constraints, one of them being a limit on the number of items: $\sum_{j\in\mathcal{N}} w_j x_j \leq C, \sum_{j\in\mathcal{N}} x_j \leq k$. The 1.5KP problem is also NP-hard.dKP problems with correlation between items have been studied in [21], where the items were partitioned into classes and up to one item per class was picked. In our case, a class is the set of all prefixes covering a certain address. Each item (prefix) can belong simultaneously to any number of classes, from one class (/32 address) to all classes (/0 prefix). To the best of our knowledge, we are the first to tackle the case where the items belong to classes that are not a partition of the set of items. In summary, the special structure of filtering problems, *i.e.*, the hierarchy and overlap of candidate prefixes, leads to novel variations of dKP that could not be solved using existing methods.

VI. Conclusion

In this paper, we introduced a formal framework for studying filtering problems rooted at the theory of the knapsack problem and providing a novel extension of it. Within this framework, we formulated four practical problems, and we designed optimal yet low-complexity (linear in the input size) algorithms that solve them. We also highlighted connections between the different problems: at the heart of all problems lies BLOCK-SOME; BLOCK-ALL and FLOODING are special instances for specific assignment of weights, and DIST-FLOODING can be decomposed into several independent FLOODING problems. Finally, we did simulations using Dshield traces; a key insight was that our algorithms can exploit the spatial clustering that is inherent in real blacklists.

There are several directions for future work. We plan to extend the framework to dynamically update the filtering rules as blacklists change over time, combine source- with destination-based filtering, deal with adversarial scenarios, and study the interaction between filtering and detection mechanisms. We will also provide a more extensive experimental evaluation, which was not the focus of this paper.

References

[1] Cisco Systems, White Paper, "Understanding acl on catalyst 6500 series switches," *www.cisco.com/en/US/products/hw/switches/ps708/products_white_ paper09186a00800c9470.shtml.*

[2] Cisco Systems, White Paper, "Protecting your core: Infrastructure protection access control lists," *http://www.cisco.com/en/US/tech/tk648/tk361/technologies_white_paper09186a00801a1a55.shtml.*

[3] Dshield *http://www.dshield.org/*

[4] H.Kellerer,U.Pferschy,D.Pisinger,"Knapsack Problems", Springer,2004.

[5] G. Varghese, "Network Algorithmics," Morgan Kaufmann, 2005.

[6] F.Soldo, A.Markopoulou, K. Argyraki, "Optimal Filtering of Malicious IP Sources", *Technical Report arXiv:cs.NI/0811.3828*, Nov. 2008

[7] G.V. Gens and E.V. Levner, "Computational complexity of approximation algorithms for combinatorial problems", *in Mathematical Foundations of Computer Science, LNCS*, vol. 74, pp.292-300, Springer 1979.

[8] D.P. Bertsekas, "Non linear programming," *Athena Scientific*, 2003.

[9] Z.Chen, C.Ji, P.Barford, "Spatial-Temporal Characteristics of Internet Malicious Sources," *in IEEE INFOCOM (Mini-Conf.)*, April 2008.

[10] E. Kohler, J. Li, V. Paxson, and S. Shenker, "Observed structure of addresses in IP traffic,"*IEEE/ACM ToN 14(6)*, pp.1207-1218,Dec.2006.

[11] K. Argyraki, D.R.Cheriton, "Active Internet Traffic Filtering: Real-time Response to DoS Attacks", *in Proc. of USENIX Security 2005*.

[12] X. Liu, X. Yang, Y. Lu, "To Filter or to Authorize: Network-Layer DoS Defense Against Multimillion-node Botnets," *in ACM SIGCOMM 2008*, Seattle, Aug.2008.

[13] R. Beverly, S. Bauer, "The Spoofer Project: Inferring the Extent of Internet Source Address Filtering on the Internet,"*in SRUTI 2005*.

[14] P.Ferguson,D.Senie,"Network Ingress Filtering: Defeating DoS Attacks which employ IP Source Address Spoofing", *RFC 2827*, May 2000.

[15] D.Andersen,H.Balakrishnan,N.Feamster,T.Koponen,D.Moon,S.Shenker, "Accountable Internet Protocol (AIP)," *in ACM SIGCOMM 2008*, Seattle, Aug. 2008.

[16] X. Liu, A. Li, X. Yang, D. Wetherall, "Passport: Secure and Adoptable Source Authentication, " *in Proc. USENIX/ACM NSDI 2008*.

[17] G. Pack, J. Yoon, E. Collins, C. Estan, "On Filtering of DDoS Attacks Based on Source Address Prefixes," *in Proc. of SecureComm*, Aug. 2006.

[18] E. Al-Shaer et al., "Firewall Policy Advisor Project", De Paul Univ. *http://www.mnlab.cs.depaul.edu/projects/SPA/*

[19] F. Soldo, K. El Defrawy, A. Markopoulou, B. Krishnamurthy, and K. v.d. Merwe, "Filtering sources of unwanted traffic" *in Proc. of Inf. Theory and Applications Workshop*, UCSD, Jan. 2008.

[20] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger, "Approximation algorithms for knapsack problems with cardinality constraints", *European Journal of Operational Research*, 123:333-345, 2000

[21] A. Bagchi, N.Bhattacharyya, N.Chakravarti, "LP relaxation of the two dimensional knapsack problem with box and GUB constraints", *European Journal of Operational Research*, Elsevier, 1994