Математические структуры и моделирование
2011, вып. 24, с. 102–106

УДК 004.056.5:004.021

ОПТИМИЗАЦИЯ СПИСКОВ ДОСТУПА НА ОСНОВЕ ОБРАБОТКИ ИНТЕРВАЛЬНЫХ ДЕРЕВЬЕВ

А.В. Зверев, Д.Н. Лавров

В статье рассматривается задача оптимизации списков доступа сетевых маршрутизаторов. Описывается алгоритм оптимизации, использующий интервальные деревья, и их структура. Приводятся результаты работы алгоритма.

Введение

Списки доступа являются одним из видов межсетевого экранирования. Списки доступа представляют собой упорядоченное множество правил, каждое из которых разрешает, либо запрещает прохождение сетевых пакетов определенного типа. При поступлении очередного сетевого пакета на интерфейс маршрутизатор поочередно сверяет каждое правило списка доступа с полученным пакетом до нахождения соответствия. Данный процесс повторяется для каждого пакета. Очевидно, что скорость обработки списка доступа напрямую влияет на скорость работы сети.

Существует два основных метода увеличения скорости обработки списков доступа. Первый метод предполагает перестановку правил. Его описание можно найти в [1,2]. Второй метод, который и будет рассмотрен далее, заключается в удалении избыточных правил списка доступа.

1. Алгоритм оптимизации

В качестве алгоритма оптимизации для проекта был выбран optimise из [3] (рис. 1). Данный алгоритм использует многомерные интервальные деревья в качестве структуры данных для представления списков доступа и реализует второй из методов их оптимизации.

Каждое правило списка доступа можно представить следующим образом:

$$ACE = \{[a_1, a'_1], ..., [a_n, a'_n]\}.$$

Каждый интервал $[a_i, a'_i]$ соответствует определенному диапазону в записи списка доступа. Например, диапазону IP-адресов отправителя. Используя

Омский государственный университет им Ф.М. Достоевского
E-mail: chokis@mail.ru, lavrov@omsu.ru

```
1.for each rule y in L
2.      S_p = I_p.query (y);
3.      S_d = I_d.query (y);
4.      let a = тип y; // permit или deny

5.      if any rule in S_a contains y
6.          continue; // избыточное правило (1)

7.      if any rule in S_!a contains y
8.          continue; // несовместимое правило

9.      for each rule x in S_a that is contained by y
10.         if (!existsDependedRule(x, S_!a))
12.             I_a.remove (x); // избыточное правило (2)

13.     for each rule x in S_a that is overlap or adjacent y
14.         if (!existsDependedRule(x, S_!a)))
16.         I_a.remove (x);
17.         y = merge(x, y); // соединение правил

18.     I_a.insert (y);

19. build L_result using I_p and I_b
20. return L_result
```

Рис. 1. Алгоритм optimise

данное представление, список доступа можно преобразовать в многомерное интервальное дерево, позволяющее осуществлять быстрый поиск (за время $O(n \log^d n)$) нужных диапазонов. Алгоритм удаляет избыточные и несовместимые правила и объединяет соседние и перекрывающиеся. Будем использовать определения для избыточных, несовместимых, соседних и перекрывающихся правил из [3].

Рассмотрим два интервала $M = [m, m']$ и $N = [n, n']$. Возможны три случая.

1. M и N пересекаются, если $(m \leqslant n \leqslant m') \wedge (m \leqslant n' \leqslant m') \wedge (n \leqslant m \leqslant \leqslant n') \wedge (n \leqslant m' \leqslant n')$.

2. M содержит N, если $(m \leqslant n) \wedge (n' \leqslant m')$.

3. M и N соседние, если $(n = m' + 1) \vee (m = n' + 1)$.

Пусть $X = [x_1, x_2, \ldots, x_d]$ и $Y = [y_1, y_2, \ldots, y_d]$ — правила списка доступа, X предшествует Y и X_p,X_d означают правила типов $permit$ и $deny$ соответственно. Тогда:

1) X, Y пересекаются, если $\forall i : x_i, y_i$ пересекаются;

2) X содержит Y, если $\forall i : x_i$ содержит y_i;

3) X и Y перекрывающиеся, если $\exists! i$, такое, что x_i пересекает y_i , для всех остальных i: $x_i = y_i$;

4) X и Y соседние если $\exists! i$, такое, что x_i соседний с y_i , для всех остальных i: $x_i = y_i$.

Определение 1. Правило Y избыточно по отношению к X, если

1) X_p содержит Y_p;

2) X_d содержит Y_d;

3) Y_p содержит X_p и $\neg\exists$ правила Z_d между X_p и Y_p, такого, что X_p пересекает Z_d и X_p не содержит Z_d;

4) Y_d содержит X_d и $\neg\exists$ правила Z_p между X_d и Y_d, такого, что X_d пересекает Z_p и X_d не содержит Z_p.

Определение 2. Правило Y несовместимо с X, если

1) X_p содержит Y_d;

2) X_d содержит Y_p.

Избыточные и несовместимые правила могут быть удалены без влияния на политику безопасности.

Определение 3. Правило X может быть объединено с Y, если

1) X_p и Y_p соседние либо пересекающиеся и $\neg\exists Z_d$ между X_p и Y_p, такого, что Z_d пересекает X_p и X_p не содержит Z_d.

2) X_d и Y_d соседние либо пересекающиеся и $\neg\exists Z_p$ между X_d и Y_d, такого, что Z_p пересекает X_d и X_d не содержит Z_p.

В качестве многомерного интервального дерева была выбрана структура из [4]. Это красно-чёрное дерево, каждый узел которого содержит:

1) Точку p — ключ, по которому ведётся поиск;

2) Упорядоченный линейный список диапазонов;

3) Ссылки на правое, левое поддеревья и родителя.

Каждый диапазон, в свою очередь, содержит ссылку на интервальное дерево следующего уровня. Например, для группы правил с одинаковым диапазоном IP-адресов отправителя 192.168.0.0 255.255.255.0 создается интервальное дерево, содержащее все диапазоны адресов получателя, указанные в данной группе правил, ссылка на которое хранится в диапазоне 192.168.0.0 255.255.255.0 узла дерева предыдущего уровня. Диапазоны в линейном списке упорядочиваются по начальным и конечным точкам. Следует отдельно отметить особенность объединения интервалов IP-адресов. Помимо соблюдения условий определения 3, для возможности объединения двух диапазонов сетевых адресов необходимо

также равенство длин диапазонов. Следовательно, в результате работы алгоритма будут объединяться диапазоны сетевых адресов одинаковой длины. Рассмотрим четыре различных диапазона сетевых адресов отправителя четырёх правил списка доступа, остальные параметры которых совпадают. Обозначим диапазоны как $\{1\}$, $\{2\}$, $\{3\}$ и $\{4\}$. Пусть диапазоны $\{1\}$ и $\{2\}$, $\{2\}$ и $\{3\}$, $\{3\}$ и $\{4\}$ соседние. На первом шаге работы алгоритма произойдёт добавление диапазона $\{1\}$ в интервальное дерево. Далее алгоритм рассмотрит диапазон $\{2\}$ и произведёт слияние с диапазоном $\{1\}$. Полученный диапазон обозначим $\{1,2\}$. Следующим шагом алгоритм рассмотрит диапазон $\{3\}$, но не сможет произвести объединение с $\{1,2\}$ ввиду неравности длин. И, наконец, алгоритм рассмотрит диапазон $\{4\}$ и произведет объединение с $\{3\}$, получив диапазон $\{3,4\}$. Таким образом получим список доступа, содержащий два правила с диапазонами $\{1,2\}$ и $\{3,4\}$ соответственно. Очевидно, что полученный список доступа не является оптимальным, т.к. существует возможность объединения диапазонов $\{1,2\}$ и $\{3,4\}$. Существует два способа решения данной проблемы. Первый заключается в повторном запуске алгоритма на полученном списке доступа. Второй способ предполагает изменение в структуре алгоритма таким образом, чтобы каждый новый диапазон, получившийся в результате объединения, рассматривался алгоритмом повторно. В проекте был реализован второй способ.

2. Результаты работы алгоритма

В качестве списков доступа использовались части одного большого списка доступа, содержащего 10 тыс. правил. Как видно из таблицы 1, результат оптимизации может существенно отличаться. Это зависит, в первую очередь, от самого списка доступа, т.е. от количества избыточных, несовместимых, соседних и пересекающихся правил.

Таблица 1. Результаты работы алгоритма

Количество правил ACL	500	673	938	664	744	1028	547	403	599
Удалено	286	284	107	28	306	156	109	164	109
Удалено в %	57.2	42.2	11.41	4.22	41.13	15.18	19.93	40.69	18.2

Заключение

Рассмотренный алгоритм оптимизации позволяет сократить используемые списки доступа за счет удаления избыточных и несовместимых правил и объединения соседних и перекрывающихся, что позволит повысить скорость обработки пакетов маршрутизатором. Временная сложность алгоритма оценивается как $O(n \log^d n)$.

Литература

1. V. Grout and J. McGinn. Optimisation of Policy-Based Internet Routing using Access Controll Lists // Proceedings of the 9th IFIP/IEEE Symposium on Integrated Network Management. 2005. URL: http://www.newi.ac.uk/groutv/Papers/IEEE_IM_ACLs.pdf 16.01.2011).
2. Ibrahim M. Al Abdulmohsin. Techniques and Algorithms for Access Control List Optimization // Computers & electrical engineering. 2009. Vol. 35, Issue 4. P. 556-566. URL: http://www.sciencedirect.com/science/journal/00457906 (дата обращения: 11.01.2011).
3. Jiang Qian, Susan Hinrichs, and Klara Nahrstedt. ACLA: A Framework for Access Control List (ACL) Analysis and Optimization // Proceedings of the IFIP TC6/TC11 International Conference on Communications and Multimedia Security Issues of the New Century. 2001. P. 4.
4. Haibin Lu, Sartaj Sahni. Enhanced Interval Trees for Dynamic IP Router- Tables // IEEE Transactions on Computers. 2004. Vol. 53, Issue 12. P. 1615-1628. URL: http://www.cise.ufl.edu/ sahni/papers/interval.pdf (дата обращения: 11.01.2011).