УДК 004.457

Савенко Д.В.[1]

[1] Новосибирский Государственный Университет.

Построение оптимального дерева правил iptables.

В настоящее время в подавляющем большинстве локальных сетей фильтрация трафика проводится вручную администраторами путем создания правил для межсетевых экранов. Нас заинтересовала задача автоматизации этого процесса — автоматическая фильтрация, использующая собранную ранее статистику передачи пакетов для данной локальной сети. Такой подход ценен не только тем, что облегчает работу администратора, но и тем, что дает возможность обнаружения атак и вирусов ранее неизвестных типов, а также неисправностей и нецелевого использования сети.

Первым этапом работы созданной нами программы является построение обучающей выборки, т.е. сбор информации о нормальной активности в сети. Для этого специальный модуль должен быть установлен на шлюзе, через который проходят пакеты локальной сети. Программа работает на уровне соединений — потоков пакетов между клиентом (инициатором соединения) и сервером. Сборщик создает базу данных с информацией о типах устанавливаемых за время его работы соединений. В дальнейшем нормальным считается пакет, принадлежащий какому-либо из этих типов, а аномальным — пакет, который нельзя отнести ни к одному из них.

Задача программы — построить набор логических решающих правил (ЛРП, [1]), которые пропускали бы нормальные пакеты и задерживали аномальные. Причем, учитывая высокие нагрузки на шлюзах локальных сетей, необходимо минимизировать среднее время прохождения пакета через эти правила. Поэтому нельзя просто перечислить параметры всех соединений в качестве правил, т.к в этом случае время прохождения пакета оценивается как $O(N)$, где N — количество типов соединений.

Метод, используемый в программе, основан на алгоритме DW [2] и сокращении перебора методом наращивания «лучшего к лучшему» [3]. Строится дерево, каждой ветви которого соответствует определенное ЛРП, имеющее вид конъюнкции элементарных высказываний (ЭВ) о признаках объектов вида $X(a) = x$, либо $X(a) \le x$, где x — это фиксированное пороговое значение. Дополнениями к ним служат высказывания вида $X(a) \ne x$ и $X(a) > x$. Дерево дихотомическое, каждому

«левому» ребру приписано какое-то ЭВ, а «правому» ребру — его дополнение. Таким образом, любой путь из вершины дерева к листу можно трактовать как конъюнкцию ЭВ и дополнений к ним. Далее по алгоритму DW мы должны были бы пропускать все объекты обучающей выборки через дерево, и каждый лист приписывать тому образу, объектов которого в листе накопилось больше. Наши образы — это нормальный и аномальный трафик. Проблема в том, что обучающая выборка состоит только из нормальных объектов. Поэтому в нашей модификации алгоритма лист признается нормальным, если в него попал хотя бы один объект из обучающей выборки, и аномальным — если не попало ни одного объекта. Перебор пороговых значений для ЭВ ведется только по тем значениям параметров, которые встречались в обучающей выборке. Такой подход обеспечивает точное отделение нормальных объектов от аномальных. Иными словами, мы можем быть уверены, что если в лист дерева попал один или несколько нормальных объектов из обучающей выборки, то аномальные объекты (которых мы не знаем) в этот лист попасть не могут. И наоборот, если в лист не попало ни одного нормального объекта, то туда могут попасть только аномальные объекты. Также благодаря этому существенно сокращается перебор, т.к. пространство значений некоторых параметров достаточно велико, в то же время число реально используемых в обучающей выборке значений обычно гораздо меньше.

Чтобы удовлетворить требование минимизации времени прохождения пакетов через дерево, при построении очередного узла мы выбираем то ЭВ, которое обеспечивает наиболее равномерное деление текущего подмножества объектов обучающей выборки. То есть, минимальной должна быть величина $|N_E - N_D|$, где E и D — ЭВ и дополнение к нему, а $N_E = \sum_{i \in E} N_i$, где N_i — это количество появлений объекта i во время построения обучающей выборки (N_D — аналогично). Это гарантирует, что чем чаще пакет появляется в сети, тем быстрее он пройдет через дерево правил — что и было главным критерием оптимальности при разработке программы.

Результатом работы над проектом является программа на языке C++ для операционной системы Linux, которая строит дерево ЛРП, а затем переводит его в формат правил iptables [4], стандартный для Linux-систем. Программа поддерживает протоколы TCP, UDP и ICMP [5]. Параметры соединений различаются в зависимости от протокола, например, для TCP это IP-адреса клиента и сервера, а также номер порта сервера (номер порта клиента не учитывается — это случайное число). В данный

момент программа находится на стадии тестирования разработчиками. Обучающая выборка для тестирования берется путем мониторинга локальной сети студенческого городка НГУ. Серьезные опасения у разработчиков вызывало то, что программа может генерировать слишком много правил. Если бы количество правил росло слишком быстро с ростом размера обучающей выборки, применимость программы в реальных условиях сводилась бы к нулю. Но на предварительных тестах (рис. 1) мы получили нечто похожее на насыщение — количество правил колебалось в пределах достаточно небольших значений ($80-95$), почти независимо от размера выборки. Это позволяет надеяться на хорошие характеристики производительности в реальных условиях.

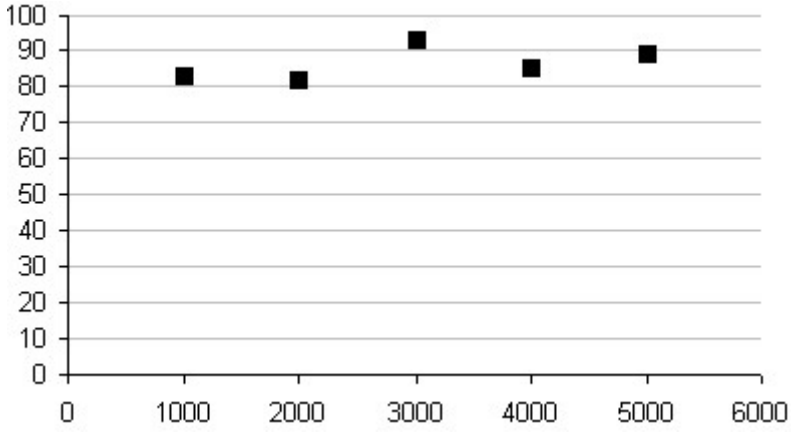

Рис. 1. График зависимости количества правил (ось ординат) от размера обучающей выборки (ось абсцисс).

СПИСОК ЛИТЕРАТУРЫ

1. *Лбов Г.С.* Методы обработки разнотипных экспериментальных данных. — Новосибирск: Наука, 1981.
2. *Загоруйко Н.Г.* Прикладные методы анализа данных и знаний. — Новосибирск: ИМ СО РАН, 1999.
3. *Барабаш Ю.Л., Варский Б.В., Зиновьев В.Т.* Автоматическое распознавание образов. — Киев: КВАИУ, 1963.
4. *Немет Э., Снайдер Г., Сибасс С., Хейн Трент Р.* UNIX: руководство системного администратора. Для профессионалов. 3-е изд. — СПб.: Питер, 2002.
5. *Олифер В.Г., Олифер Н.А.* Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 2-е изд. — СПб.: Питер, 2003.