УДК 004.042

Шыхалиев Р.Г.
Институт Информационных Технологий НАНА, Баку, Азербайджан
ramiz@science.az

ОБ ОДНОМ АЛГОРИТМЕ МОНИТОРИНГА СЕТЕВЫХ ЧЕРВЕЙ С ПЕРМУТАЦИОННЫМ СКАНИРОВАНИЕМ

В последние годы появились различные стратегии заражения сетевых червей и увеличилась скорость их распространения. Поэтому для обнаружения сетевых червей, в частности пермутационных червей, высокоскоростной мониторинг и анализ сетевого трафика в режиме реального времени является важными. Однако из-за появления вычислительных трудностей и проблем с хранением потоков данных решение этой задачи с использованием детерминированных алгоритмов становится очень трудным. Поэтому в статье для мониторинга сетевого трафика предлагается использовать рандомизированные потоковые алгоритмы, в частности, метод скользящего окна, для которых требуется очень мало памяти и используется мало вычислительных ресурсов.

Ключевые слова: *сетевые черви, пермутационное сканирование, мониторинг сетевого трафика, скользящее окно.*

1. Введение

В последние годы, намного увеличились сложность механизмов заражения и скорость распространения сетевых червей. Сегодня, сетевые черви способны за очень короткое время заразить миллионы хостов подключенных к Интернету, и в результате нанести большой ущерб. При этом для распространения сетевых червей используются различные стратегии сканирования [1], такие как сканирование на основе предпочтения локальных адресов, топологическое сканирование, сканирование по заранее составленным спискам уязвимых узлов, случайное сканирование, последовательное сканирование, сканирование на основе пермутации, частичное сканирование.

При пермутационном сканировании все экземпляры червя совместно используют общее пространство IP-адресов перестановленных псевдослучайно [2]. Такая перестановка может быть сгенерирована с использованием любой блочной шифры длиной 32 бит, с предварительно выбранным ключом. При таком подходе маловероятно, что разные экземпляры червя будут выбирать одну и ту же жертву-хоста и не будут тратить время на сканирование одного и того же хоста несколько раз. Каждый зараженный из заранее составленных списков уязвимых хостов или зараженный в результате сканирования локальных адресов подсети хост, сразу в поисках новых уязвимых хостов начинает сканирование на пространстве пермутации. При инфицировании черви с пермутационным сканированием начинают сканирование со случайной точки. Модель распространения червей с пермутационным сканированием основывается на исходе достижении уязвимых хостов сообщения сканирования, т.е. когда сообщения сканирования достигают уязвимых хостов, то количество зараженных, активных и удаленных хостов изменяется [3]. Таким образом, когда хост заражается червем, то он пытается распространить этот червь другим хостам, чтобы заразить их, и тем самым генерирует очень большое количество сетевых соединений.

Поскольку активные хосты способствуют быстрому распространению червей с пермутационным сканированием, то их раннее определение является очень важным. В частности, раннее определение активных хостов позволяет обнаружить и предотвратить распространение червей с пермутационным сканированием и в результате минимизировать нанеенный ущерб. Однако решение этой задачи невозможно без

постоянного и быстрого мониторинга и анализа сетевого трафика в режиме реального времени. Для этого предлагается алгоритм мониторинга и анализа сетевого трафика, который позволит в режиме реального времени определить хосты, которые связываются с большим числом различных хостов, то есть отправляют один и тот же пакет большому числу различных хостов. Суть предложенного алгоритма заключается в том, что в течение определенного короткого периода времени из заданного потока пар IP-адресов хостов (s, d), имеющегося в сетевом трафике, необходимо из s определить такие IP адреса, которые имеют связи с большим числом различных IP-адресов из d, т.е. состоят в паре с большим числом IP-адресов из d, где s и d соответственно являются IP-адресами хостов-источников и хостов-назначений.

Однако из-за появления вычислительных трудностей и проблем с хранением потоков данных решение этой задачи с использованием детерминированных алгоритмов становится очень трудным. Это в основном связано с тем, что объем и скорость непрерывно поступающих трафиков в современных компьютерных сетях очень высоки и может содержать миллионы потоков IP-пакетов. Поэтому для решения этой задачи предлагается использовать рандомизированные потоковые алгоритмы, которые используют меньше вычислительных ресурсов и памяти, чем детерминированные алгоритмы. В частности, в качестве потокового алгоритма предлагается использовать метод скользящего окна [4, 5], который использует очень ограниченное пространство памяти для хранения данных и меньше вычислительных ресурсов из-за обработки данных за один проход.

2. Модель распространения сетевых червей с пермутационным сканированием

В работе [3] показано, что сетевые черви с пермутационным сканированием посредством шифрования могут осуществлять перестановку пространства IP-адресов в виртуальное, так называемое пермутационное кольцо, и совместно использовать общее пространство IP-адресов, перестановленных псевдослучайно. А также авторами предлагается математическая модель, которая точно характеризует распространение сетевых червей с пермутационным сканированием.

Пусть имеется пространство IP-адресов размера N, с общим числом уязвимых хостов V, скоростью сканирования червя r и v – число уязвимых хостов в заранее составленном списке уязвимых хостов. А также для уязвимых хостов определяются следующие классы: незараженные, зараженные, активные, удаленные, эффективные, неэффективные и зарождающиеся и соответственно обозначаются как u, i, a, s, x, y и α, а популяции этих классов во времени t, соответственно обозначаются как u (t), i(t), a(t), s(t), x(t), y(t) и α(t).

Формально, модель распространения червей с пермутационным сканированием – это отображение во времени множества уязвимых хостов V на зараженные – i(t), активные – a(t) и удаленные – s(t). Причем количество зараженных, активных и удаленных хостов изменится тогда и только тогда, когда сообщения сканирования червя достигнут какого-либо уязвимого хоста. Поэтому модель распространения червей с пермутационным сканированием основывается на исходе, достижении сообщения сканирования уязвимых хостов, т.е. когда сообщения сканирования достигают уязвимых хостов, количество зараженных, активных и удаленных хостов изменяется.

Основная идея модели, предложенной а работе [3], заключается в установлении изменений во времени популяций $i(t)$, a(t), s(t), x(t), y(t) и α(t). Для этого вычисляются значения $di(t)$, da(t), ds(t), dx(t), dy(t) и dα(t) за бесконечно малое время dt. Причем $u(t) + i(t) = 1$, $i(t) = a(t) + s(t)$ и $a(t) = x(t) + y(t)$.

Прежде чем построить модель распространения червей с пермутационным сканированием, предварительно были вычислены следующие величины. Пусть, f_{hit} –

число уязвимых хостов, которые, как ожидается, сообщения сканирования активного хоста достигнут за время dt. А также, за счет случайности процесса перестановки адресов, уязвимые хосты равномерно распределены на пермутационном адресном пространстве, и каждый адрес на пермутационном кольце имеет вероятность $\frac{V}{N}$ быть уязвимым хостом. Активный хост сканирует $r \times dt$ адресов в течение периода dt, следовательно, $f_{hit} = r \times dt \times \frac{V}{N}$. При этом необходимо подчеркнуть, что уязвимые хосты, которые достигают сообщения сканирования, могут включать как вновь (новые) зараженные, так и уже (старые) зараженные хосты.

Для случая, когда сообщения сканирования эффективного хоста достигнут уязвимого хоста, через $f_{new}(t)$ и $f_{old}(t)$ обозначаются вероятности того, что уязвимый хост может быть новозараженным или старозараженным. Однако необходимо подчеркнуть, что сообщения сканирования эффективного хоста могут достигнуть только двух типов уязвимых хостов: незараженных и зараженных хостов. На момент времени t имеется $V(1 - i(t))$ незараженных уязвимых хостов и $V(x(t) - \alpha(t))$ хостов. Таким образом, вероятности новых и старых заражений соответственно будут выражаться формулами (1) и (2):

$$f_{new}(t) = \frac{V(1-i(t))}{V(1-i(t))+V(x(t)-\alpha(t))} = \frac{(1-i(t))}{(1-i(t))+(x(t)-\alpha(t))}\ , \quad (1)$$

$$f_{old}(t) = 1 - f_{new}(t) = \frac{(x(t)-\alpha(t))}{(1-i(t))+(x(t)-\alpha(t))}\ . \quad (2)$$

После того как хост заражается, он переходит на случайное новое место, чтобы начать сканирование хостов. Пусть $f_{ineff}(t)$ и $f_{eff}(t)$ соответственно являются вероятностями того, что новый зараженный хост будет неэффективным или эффективным, и определяются следующим образом: $f_{ineff}(t) = \frac{Vi(t)-V(x(t)-\alpha(t))}{V}$ и $f_{eff}(t) = 1 - f_{ineff}(t)$.

Наконец, в результате вычислений значений $di(t)$, $d\mathrm{a}(\mathrm{t})$, $d\mathrm{s}(\mathrm{t})$, $d\mathrm{x}(\mathrm{t})$, $d\mathrm{y}(\mathrm{t})$ и $d\alpha(\mathrm{t})$ за бесконечно малое время dt была получена модель распространения червей с пермутационным сканированием, состоящая из системы дифференциальных уравнений, которые в совокупности характеризуют распространение червей с пермутационным сканированием, и эта модель более подробно описана в работе [3].

3. Алгоритм мониторинга сетевых червей с пермутационным сканированием

В этом разделе мы представляем формальное определение проблемы, а затем описываем алгоритм определения активных хостов.

Как было сказано выше, модель распространения червей с пермутационным сканированием основывается на исходе, достижении сообщения сканирования уязвимых хостов, т.е. когда сообщения сканирования достигают уязвимых хостов, количество зараженных, активных и удаленных хостов изменяется. Причем активный хост сканирует $r \times dt$ адресов в течение периода dt, количество которых может быть в несколько тысяч. Например, зараженные Slammer червями хосты за секунду отправляют до 26000 сообщений сканирования [6]. Поэтому для раннего обнаружения червей с пермутационным сканированием необходимо проведение быстрого мониторинга, который позволит в режиме реального времени определить активные хосты.

3.1. Формальная постановка задачи

Пусть дан сетевой трафик, который смоделирован в виде потока, состоящего из пары адресов источник-назначение $(\mathrm{s},\mathrm{d},$ поступающих последовательно (причем рассматриваются потоки, в которых $d_i \neq d_j$) и случайные хэш-функции [7] h_1 и h_2, которые позволят осуществлять отбор случайных равномерных выборок, состоящих из различных пар адресов источник-назначение. Здесь, хэш-функция h_1 отображает пары адресов источник-назначение (s,d) в $[0,1]$, то есть $h_1\colon (s,d) \rightarrow [0,1]$, а хэш-функция h_2 отображает адреса источников s в $[0,1]$. Причем хэш-функции h_1 и h_2 должны удовлетворять требованиям быстрого вычисления и минимальных коллизий. Таким образом, каждая пара адресов источник-назначение, а также отдельные адреса источников с равной вероятностью могут отображаться на любое значение в интервале $[0,1]$. При этом предполагается, что хэш-функции являются независимыми и их использование гарантирует то, что вероятность включения в выборку не зависит от числа появления в потоке отдельных пар адресов источник-назначение, то есть каждая пара с равной вероятностью может быть включена в выборку. А также пусть даны две хэш-таблицы T_1 и T_2, которые соответственно связаны с хэш-функциями h_1 и h_2, где хэш-таблица T_1 предназначена для обнаружения и исключения повторяющихся пар адресов источник-назначение, а хеш-таблица T_2 предназначена для подсчета количества различных адресов назнпчений, с которыми связывается каждый источник-адрес.

Поскольку в потоке могут быть пакеты, которые в заголовке имеют одни и те же пары адресов источник-назначение, то отображение, осуществляемое хэш-функциями h_1 и h_2, может привести к коллизиям [7]. При возникновении коллизии, две или более одиноковых пары адресов источник-назначение ассоциируются с одной и той же ячейкой хэш-таблиц. Поэтому в целях разрешения проблемы коллизий для каждой хэш-функции h_1 и h_2 используются хэш-таблицы с $\frac{N}{k}$ блоками, которые могут содержать записи, отображаемые хэш-функциями в одном и том же адресе в хэш-таблицах. Это позволит обнаружить и исключить повторяющиеся пары адресов источник-назначение, а также подсчитать количество различных адресов назначений, с которыми связывается каждый источник-адрес. При этом, количество блоков в хэш-таблицах определяет нижнюю границу памяти, необходимую для определения активных хостов в потоке.

Требуется определить активные хосты по последним W-пакетам, то есть определить хосты, которые связываются с большим количеством различных хостов в W-пакетах, где W является длиной скользящего окна. При этом должно использоваться меньше памяти, чем необходимо для хранения всех пар IP-адресов в текущем окне.

3.2. Описание алгоритма поиска активных хостов

Сначала определим активные хосты как хосты с адресами s, у которых в рамках окна W с N парами источник-назначение адресами хостов число связей с различными хостами из адресов d превышает некоторый заранее заданный порог k. При этом в заданном наборе пакетов N может быть N/k активных хостов и для хранения их понадобится пространство $\Omega(N/k)$, что дает нам нижнюю границу, необходимую для нахождения активных хостов.

Простым подходом к уменьшению используемой памяти является рандомизированное удаление пары адресов источник-назначение из потока, то есть из целого потока пар адресов хранится только малая часть случайно выбранной пары адресов источник-назначение. При этом основная идея заключается в том, что в выборку попадут те источники-адреса s, которые соединяются со множеством различных адресов-назначений d.

Отбор выборок из потока пар адресов источник-назначение основывается на хэшировании [7] потока пары адресов источник-назначение. Каждая входящая пара адресов источник-назначение хэшируется с помощью хэш-функции h_1, и полученные для каждой пары адресов источник-назначение хэш-значения используются для расчета индексов расположения в хэш-таблице T_1. Эти индексы используются для проверки на наличие пары адресов источник-назначение в хэш-таблице T_1. Если текущая входная пара адресов источник-назначение присутствует в хэш-таблице T_1, то алгоритм переходит к следующей входящей паре. В противном случае пара адресов источник-назначение добавляется в определенное место в хэш-таблице T_1. А хэш-таблица T_2 содержит счетчик для подсчета количества различных адресов назначений, с которыми связывается каждый источник-адрес. При этом, если текущая входная пара адресов источник-назначение не присутствует в хэш-таблице T_1, то это означает, что пара не поступала раньше, и, следовательно, счетчик для подсчета количества различных адресов назначений соответствующего источника-адреса увеличивается на одну единицу. Источники-адреса, имеющиеся в парах, хэшируются в хэш-таблицу T_2, и если источник-адрес присутствует в таблице, то его счетчик увеличивается на одну единицу. В противном случае в хэш-таблицу T_2 для подсчета нового источника-адреса добавляется новый блок. В итоге те источники-адреса считаются активными количество которых превышает некоторый заранее заданный порог k.

В скользящем окне рассматриваются только N последние пары адресов источник-назначение в которых источники-адреса связываются с различными назначениями-адресами. То есть по прибытии новой пары адресов источник-назначение, старая пара должна быть удалена из результатов. Это достигается путем добавления в хэш-таблицу T_1 временных меток n для каждой пары адресов источник-назначение, то есть каждая пара хранится вместе с временной меткой, которая может показывать время прибытия или логический номер последовательности сетевых пакетов (то есть в хэш-таблице T_1 хранится тройка (s, d, n)). По прибытии последующей пары адресов источник-назначение, у выходящей из окна пары временная метка сбрасывается. В случае появления нескольких однаковых пар в текущем окне временная метка устанавливается на последней паре. При каждом поступлении новой пары адресов источник-назначение ее временная метка становится n_{curr} и в хэш-таблице T_1 производится поиск пары с временной меткой $t_{curr} - W$ и удаляется из таблицы. После этого значения счетчика подключения источника-адреса $cont$ в хэш-таблице T_2, предназначенного для подсчета количества различных адресов назначений, с которыми связывается каждый источник-адрес, уменьшаются на одну единицу, чтобы отразить эти изменения. При этом хранятся только потоки тех пар адресов источник-назначение (s, d), для которых счетчик пакетов имеет значение большее, чем $\epsilon *$ k, ошибки фактического количества уникальных соединений источников s, где ϵ является порогом ошибки. То есть, отклонение фактического числа различных соединений каждого источника s от счетчика может быть $\epsilon *$ k.

Если подробно, алгоритм поиска активных хостов состоит из следующих шагов и фактически являются алгоритмом поиска хостов, массово сканирующих различные хосты [8 – 10] :

Шаг 1. Начало. Хэш-таблицы T_1 и T_2 пусты. В хэш-таблице T_1 хранятся тройки (s, d, n), где n является номером пары адресов источник-назначение (s, d) в потоке, а в хэш-таблице T_2 хранятся двойки $(s, count)$, где s является адресом источников, а $count$ является счетчиком подсчета количества различных адресов назначений, с которыми соединяются s. Пары адресов источник-назначение (s, d) поступают с последовательностью n (где $n = 1, 2, 3, \ldots$).

Шаг 2. Добавление в окно новой пары адресов источник-назначение (s, d). При каждом добавлении новой пары (s, d) с номером n_{curr} из окна удаляется последняя пара (s, d) с номером $n_{last} = n_{curr} - W$.

Шаг 3. Проверяется наличие тройки (s, d, n_{last}) этой пары в хэш-таблице T_1, если она существует, то удаляется.

Шаг 4. В хэш-таблице T_2 осуществляется поиск адреса источника s и при нахождении значение счетчика соединения его с различными адресами назначений d уменьшается на одну единицу, то есть двойка $(s, count)$ заменяется на $(s, count - 1)$.

Шаг 5. Если значение счетчика соединения $count$ источника s равняется нулю, то это означает, что этот источник больше не появится в выборке и двойка $(s, count)$ удаляется из хэш-таблицы T_2.

Шаг 6. Добавление в хэш-таблицу T_1 новой пары адресов источник-назначение (s, d). Если, $h_1(s, d) \geq \epsilon * k$, то проверяется наличие тройки для пары (s, d) в хеш-таблице T_1, если она существует, то заменяется тройкой (s, d, n_{curr}) и переходит к шагу 2. В противном случае в хэш-таблицу T_1 добавляется тройка (s, d, n_{curr}).

Шаг 7. В хэш-таблице T_2 осуществляется поиск адреса источника s, и если адреса источника s в хэш-таблице T_2 нет, то его счетчик соединений устанавливается на единицу, то есть в хэш-таблицу T_2 записывается двойка $(s, 1)$, так как для источника s это первое d– назначение. А если источник s в хеш-таблице T_2 имеется, то значение счетчика соединения его с различными адресами назначений d увеличивается на одну единицу.

Шаг 8. Если значение счетчика соединения источника s $count \geq k$, то источник s является активным хостом.

4. Заключение

Для обнаружения сетевых червей с пермутационным сканированием, которые могут с большой скоростью распространяться в компьютерных сетях, очень важны высокоскоростной мониторинг и анализ сетевого трафика в режиме реального времени. Однако из-за появления вычислительных трудностей и проблем с хранением потоков, решение этой задачи с использованием детерминированных алгоритмов становится очень трудным. Поэтому для мониторинга сетевого трафика требуются быстрые потоковые алгоритмы, которые используют очень малую память и мало вычислительных ресурсов. Исходя из этого, в статье для обнаружения сетевых червей с пермутационным сканированием в качестве потокового алгоритма предлагается использовать метод скользящего окна, который использует очень ограниченное пространство памяти для хранения данных и из-за обработки данных за один проход – меньше вычислительных ресурсов.

Литература

1. Smith C., Matrawy A., Chow S. and Abdelaziz B. Computer Worms: Architecture, Evasion Strategies, and Detection Mechanisms // Journal of Information Assurance and Security, 2009, no.4, pp. 69–83.
2. Weaver N. Potential Strategies for High Speed Active Worms: A Worst case Analysis, 2002 http://www.icsi.berkeley.edu/~nweaver/worms.pdf
3. Manna P.K, Shigang Chen, Ranka S.; Inside the Permutation-Scanning Worms: Propagation Modeling and Analysis, IEEE/ACM Transactions On Networking, June 2010, vol. 18, no. 3, pp. 858–870.
4. Mayur Datar and Rajeev Motwani. The sliding-window computation model and results. Data Streams The Kluwer International Series on Advances in Database Systems, 2007, vol.31, pp.149–167.
5. Aggarwal C. (editor). Data Streams: Models and Algorithms, Springer Verlag, 2007. 354 p.

6. Moore D., Paxson V., Savage S., Shannon C., Staniford S. and Weaver N. Inside the slammer worm // Security and Privacy Magazine, July/August 2003, pp. 33–39.
7. Knuth D.E. The Art of Computer Programming, Volume 3: Sorting and Searching, 2nd ed. Addison-Wesley, 1998, ISBN 0-201-89685-0, 800 p.

8. Venkataraman S., Song D.X., Gibbons P.B., and Blum A. New streaming algorithms for fast detection of superspreaders / Proceedings of the Network and Distributed System Security Symposium, 2005, San Diego, California, USA.
9. Bandi N., Agrawal D., El Abbadi A. Fast Algorithms for Heavy Distinct Hitters using Associative Memories / 7th International Conference on Distributed Computing Systems (ICDCS 2007), yune 25-29, 2007, Toronto, Ontario, Canada, pp. 247–256.
10. Locher T., Finding Heavy Distinct Hitters in Data Streams / 23rd ACM Symposium on Parallelism in Algorithms and Architectures, San Jose, California, USA, June 2011, pp. 299–308.

UOT 004.042
Şıxəliyev Ramiz H.
AMEA İnformasiya Texnologiyaları İnstitutu, Bakı, Azərbaycan
ramiz@science.az
Permutasion şəbəkə soxulcanlarının monitorinqi alqoritmi haqqında
Son illər şəbəkə soxulcanlarının yoluxmasının yeni strategiyaları meydana çıxıb və onların yayılması sürəti artıb. Buna görə də şəbəkə soxulcanlarının, xüsusilə də permutasion soxulcanların aşkarlanması üçün şəbəkə trafikinin real zaman rejimində yüksək sürətli monitorinqi və analizi vacibdir. Lakin hesablama çətinliyinin və verilənlər axının yadda saxlanması problemi yarandığı üçün bu məsələnin determinik alqoritmlərin istifadəsi ilə həlli çox çətinləşir. Buna görə də məqalədə şəbəkə trafikinin monitorinqi üçün randomizə olunmuş axın alqoritmlərinin, xüsusilə çox kiçik yaddaş tələb edən və az hesablama resursu istifadə edən sürüşən pəncərə metodunun istifadəsi təklif olunur.
***Açar sözləri**: şəbəkə soxulcanları, permutasion skanlanlaşdırma, şəbəkə trafikinin monitorinqi, sürüşən pəncərə.*

Ramiz H. Shikhaliyev
Institute of Information Technology of ANAS, Baku, Azerbaijan
ramiz@science.az
About one monitoring algorithm of permutation worms
In recent years different strategies have emerged for increasing their rate of proliferation of worms. Therefore, for the detection of worms, particularly permutation worms, high speed monitoring and analysis of network traffic in real time is important. However, due to the emergence of computational difficulties and problems with data flow storage, the solution of this problem using a deterministic algorithm becomes very difficult. Therefore, we propose a method for monitoring network traffic through the use of randomized streaming algorithms, in particular a sliding window mechanism, which requires very little memory and computational resources.
***Key words:** worms, permutation scanning, network traffic monitoring, sliding window.*