РАЗРАБОТКА АЛГОРИТМА ПОИСКА АНОМАЛИЙ В ПОВЕДЕНИИ ПРОГРАММНОГО ПРОЦЕССА С ПОМОЩЬЮ СИНТЕЗА АВТОМАТА ЭВОЛЮЦИОННЫМ АЛГОРИТМОМ

Лабутина Е. А. магистрант кафедры безопасности киберфизических систем Университета ИТМО, jenialab@gmail.com

Аннотация

В работе описывается алгоритм анализа трасс системных вызовов. Предложен алгоритм на основе построения автомата и его дальнейшей оптимизации с целью снижения числа состояний.

Введение

В соответствии с пунктом 14 Доктрины информационной безопасности возрастает число преступлений в финансовой сфере с нарушением конституционных прав и свобод человека и гражданина, в том числе в части, касающейся неприкосновенности частной жизни, личной и семейной тайны, при обработке персональных данных с использованием информационных технологий. [3].

В большинстве случаев для достижения этих целей преступники используют компьютерные вирусы. И не всегда антивирусные программы способны предотвратить утечку конфиденциальной информации или кражу денежных средств. Современные антивирусы используют методы на основе сигнатур. Но таким образом очень сложно обнаружить 0day уязвимости, против которых еще не разработаны защитные механизмы. Однако даже уязвимости 1-го дня, которые уже были опубликованы и закрыты производителем ПО, представляют опасность из-за того, что администраторы не успевают своевременно обновлять ПО. Поэтому очень важно вовремя обнаружить вирус. Одним из наиболее перспективных направлений в области обнаружения вредоносных программ является поиск аномалий в работе процессов ОС.

Аномалия - это период работы программы, в течение которого она производит действия, которые не наблюдались ранее. Модель программы – это способ описания действий процесса ОС за некоторый ограниченный период его работы.

FSA (Finite-State Machine)

Модель нормального поведения для обнаружения аномалий может демонстрировать автомат конечных состояний (далее - FSA). Автомат может “захватить” такие программные структуры как ветвления и петли, а также бесконечное количество последовательностей произвольной длины в конечном хранилище.

Модель FSA состоит из системных вызовов программы. Сбор системных вызовов производился с помощью программы, описанной в работе [2]. Автомат строился по массиву, где каждый элемент массива – номер системного вызова процесса.

Для построения детерминированного автомата применялся алгоритм А. Ахо (Alfred V. Aho) и М. Корасик (Margaret J. Corasick) [3]. Детерминированным конечным автоматом (ДКА) называется такой автомат, в котором из любого состояния по любому символу возможен переход не более, чем в одно состояние.

В этом алгоритме для каждого состояния автомата хранятся переходы только для некоторых символов входного алфавита и “переход в случае ошибки”. Если для нового входного символа существует переход из текущего состояния, то осуществляем его, если же он отсутствует, то используем “переход в случае ошибки” и еще раз проверяем наличие перехода для того же входного символа (теперь уже для нового состояния).

Для строки, состоящей из 8727 системных вызовов, получился автомат величиной более 8 миллионов состояний, построение такого автомата заняло значительное время.

Попробуем построить суффиксный автомат. Алгоритм построения автомата за линейное время описан в работе. [4] Суффиксный автомат представляет собой сжатую информацию о всех подстроках данной подстроки. Пути в суффиксном автомате соответствует строке, которая образует метки его дуг.

Количество состояний автомата для строки длиной 8727 составил чуть меньше 18 000 состояний, что значительно меньше, чем при построении алгоритмом Ахо-Корасика. Также стоит учесть, что автомат строится не только для самой строки, а также для его суффиксов. Суффикс строки A[j..] — это строка из |A|-j+1 последних символов.

Проблема в количестве состояний может возникнуть тогда, когда в строке много повторяющихся символов. Например, для строки, содержащей 100 букв «а» будет построен автомат из 101 состояния.

Построим автоматы для каждой строки отдельно и посчитаем количество состояний. Как видно из рисунка 1, это линейная зависимость.

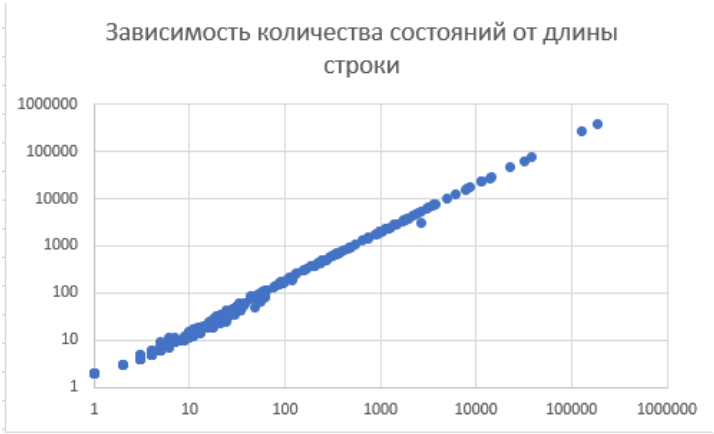

Рисунок 1: Зависимость количества состояний автомата от длины строки

Опять же причиной этого могут являться строки с большим количеством повторяющихся символов. Сделаем простейшее преобразование – возьмем 3 состояния, первые два из которых имеют одинаковый символ перехода, в третьем нет этого символа и количество переходов 2-го состояния равно одному. В таблице 1 показаны результаты данного преобразования.

Длина строки	**Количество состояний до преобразования**	**Количество состояний после преобразования**
79	141	135
309	603	562
367	716	667
8727	17428	14504
1484	2926	2764
5008	9997	9519
11268	22529	21573

Таблица 1: Зависимость длины строки и количества состояний до и после преобразования

Заключение

В документе были представлена модель нормального поведения программы в форме автомата. Автомат строился из строк, где каждый символ строки – номер системного вызова процесса.

Для построения автоматов использовалось 2 алгоритма – Ахо-Корасика и суффиксный автомат. В дальнейшем ходе работы было решено использовать суффиксный автомат, так как его построение занимает линейное время, а также количество состояний значительно меньше. Сравнение производилось на строке длиной 8727 символов. Но зависимость количества состояний от длины строки также линейна. Для автоматов было выполнено простейшее преобразование с целью уменьшения количества состояний. В среднем количество состояний уменьшилось на 7,1%. В дальнейшем планируется оптимизировать автомат таким образом, чтобы новые трассы не добавляли состояний.

Литература

1. Доктрина информационной безопасности РФ. Утв. Указом Президента РФ от 05.12.2016 № 646. Режим доступа [Электронный ресурс] / URL: http://www.kremlin.ru/acts/bank/41460/page/2, режим доступа: свободный, дата обращения: 10.05.17.
2. Баклановский М.В., Ханов А.Р., Комаров К.М., Лозов П.А. Оценка точности алгоритма распознавания вредоносных программ на основе поиска аномалий в работе процессов // Научно-технический вестник информационных технологий, механики и оптики. 2016. Т. 16. № 5. С. 823–830. doi: 10.17586/2226-1494-2016-16-5-823-830
3. Alfred V. Aho, Margaret J. Corasick. Efficient string matching: An aid to bibliographic search // Communications of the ACM. — 1975. — Т. 18, № 6. — С. 333—340. — DOI:10.1145/360825.360855.
4. A. Blumer, J. Blumer, A. Ehrenfeucht, D. Haussler, R. McConnell. Linear Size Finite Automata for the Set of All Subwords of a Word. An Outline of Results [1983]