СБОР И ПЕРВИЧНЫЙ АНАЛИЗ ДАННЫХ О ПОВЕДЕНИИ ПРОЦЕССОВ ОС

Баклановский М.В. Университет ИТМО, mb@cit.ifmo.ru,
Комаров К.М. Санкт-Петербургский государственный университет,
Лозов П.А. Санкт-Петербургский государственный университет,
Ханов А.Р. Университет ИТМО awengar@gmail.com

Аннотация

В работе рассмотрена задача анализа поведения процессов операционной системы с целью их идентификации. Рассмотрены различные способы сбора данных о работе процессов, выбран один из способов. Приведены результаты первичного анализа полученных данных с помощью алгоритма моделирования контекста PPM. Экспериментальным путем получена длина контекста, при которой потоки вызовов имеют наименьшую энтропию.

Введение

Процесс операционной системы состоит из набора потоков. Каждый процесс состоит хотя бы из одного потока. Программа стартует с точки входа и может исполняться сколь угодно долго. В процессе своей работы поток может работать лишь со своей оперативной памятью. Из-за высокой скорости работы процесса сбор данных о работе программы связан с высокими накладными расходами. Так при запуске программы в режиме трассировки скорость ее исполнения может замедлиться в сотни раз. При использовании технологии инструментирования кода, такой как PIN, удается достичь более высокой скорости [1].

Но для сбора информации о поведении процесса не обязательно знать всю трасу исполнения программы, каждую исполненную ей инструкцию. Достаточно записывать то, какие действия она произвела в системе. Чаще всего в промышленных решениях используются различные техники перехвата библиотечных вызовов (API). Обзор методов перехвата представлен в работе [2].

Не все вызовы, которые мы можем перехватить из библиотек, которые используют программы, связаны со взаимодействием с ОС. Например вызов lstrlenA из библиотеки kernel32.dll в ОС Windows считает длину строки, не обращаясь при этом к ядру ОС. Нам интересны прежде всего те из них, которые совершают изменения в объектах ядра системы – файлах, мьютексах, дескрипторах, процессах. Вызовы, которые обращаются к ядру

системы, в ОС Windows называются NativeAPI, в Linux они называются сервисами ядра. В данной работе представлен способ сбора данных о взаимодействии программы с ядром ОС Windows и первичный анализ собранных данных.

Перехват системных вызовов

Для изменения состояния операционной системы (чтение и запись в файл, работа с сетью, с другими процессами) программный процесс вызывает сервис ядра (системный вызов) с помощью команд syscall, sysenter или int XXh в процессорах Intel. С помощью процессорного механизма передачи управления, связанного с этими командами, вызывается обработчик. Он выполняется в ring-0. Его задача – найти в системной таблице SSDT адрес функции, которая вызовется для совершения запрошенной операции. Все эти функции пронумерованы. Программы, которые вызывают системные вызовы напрямую (ассемблерными командами), практически не встречаются. Как правило, используются функции из системных библиотек ОС, которые являются обертками над этими вызовами (ntdll.dll,kernel32.dll,…).

Был написан драйвер для операционных систем семейства Windows, позволяющий записывать номера системных вызовов каждого потока ОС. Более подробно можно ознакомиться в работе [3]. Он подменяет стандартный обработчик, вызываемый командой sysenter, записывает номер функции, которая была вызвана, в свой внутренний буфер и передает управление на стандартный обработчик.

Благодаря этому мы можем перехватывать активность каждого потока каждого процесса ОС с высокой скоростью, при этом учитывая только те операции, которые взаимодействуют с ядром. В текущей реализации драйвер замедляет работу процессов ОС не более чем на 10% (в случае частых обращений в ядро порядка 10 000 вызовов в секунду). На рисунке 1 приведена схема работы программы.

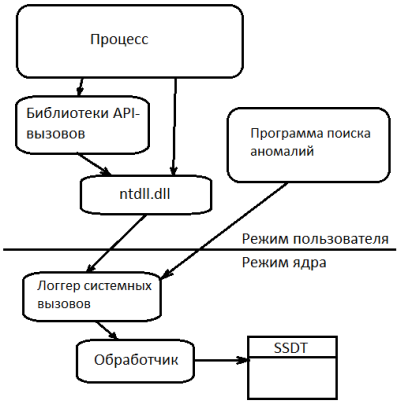

Рисунок 1. Схема взаимодействия между процессом, ядром ОС и драйвером для записи системных вызовов

Было установлено, что в ОС Windows XP системные вызовы генерируются с интенсивностью порядка 20 000 в минуту в режиме простоя, а при запуске программ и интенсивном взаимодействии с ядром ОС это число может достигать сотен тысяч. Все системные вызовы имеют обертки в виде функций, экспортируемых библиотекой ntdll.dll. Путем ее реверс-инжиниринга было установлено соответствие между именами функций и номерами системных вызовов.

Первичный анализ трасс системных вызовов

Были проведены эксперименты по записи активности программ ОС Windows XP. Были записаны трассы всех системных процессов. Наибольший интерес представляют процессы браузеров как наиболее частые объекты атак.

Потоки процессов вызывают функции системных библиотек, которые приводят к вызову целых групп системных вызовов. Для оценки влияния вероятности появления вызова в контексте была посчитана энтропия.
Теоретико-информационная энтропия – это мера информации, содержащейся в последовательности. Она рассчитывается по формуле:

$$H(x) = -\sum p(i) \log p(i)$$

p(i) – это вероятность каждого символа последовательности x. Обычно p(i) считается как 1/k, где k- число встречаемости символа.

Более интересный вариант расчета P – это вероятность символа в контексте его префикса. Одним из таких алгоритмов является PPM [4]. Обычно он используется для предсказания вероятности символа

последовательности при сжатии. Алгоритм адаптивен, он постепенно дообучается и все лучше предсказывает символы по контексту. Алгоритм учитывает префиксы до некоторой максимальной длины.

Частная энтропия – это величина, обратная вероятности события. По алгоритму PPM была рассчитана частная энтропия каждого вызова потока. На рисунке приведен график частной энтропии каждого вызова для процесса iexplore.exe (по горизонтали – номер вызова, по вертикали – частная энтропия).

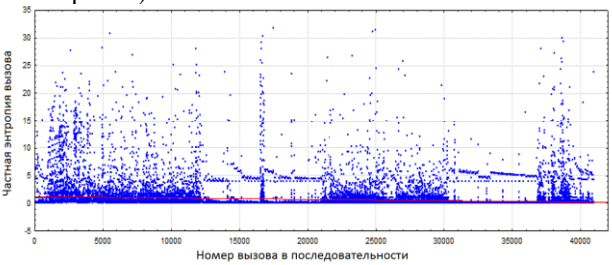

Рисунок 2. Частная энтропия каждого вызова, рассчитанная по алгоритму PPM (длина контекста 9), самый длинный поток

По графику видно, что даже в конце периода наблюдения процесс производит неожиданные для контекста системные вызовы. По значениям энтропии вызовов было посчитано среднеквадратическое отклонение σ(x) и среднее арифметическое. Выяснилось, что 1,7% вызовов имеют энтропию больше чем 3*σ(x). Во второй половине потока таких вызовов около 0,31%. Хотя модель PPM адаптируется к потоку, в потоке вызовов все равно с течением времени появляются не свойственные своему контексту номера.

Была вычислена общая энтропия H(x) последовательности при различных значениях максимальной длины контекста.

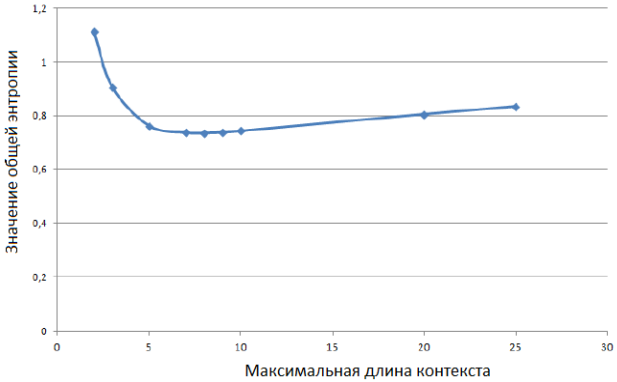

Рисунок 3. Значение общей энтропии при различных значениях максимальной длины контекста алгоритма PPM.

На рисунке изображен график зависимости общей энтропии от максимальной длины контекста в алгоритме PPM. Энтропия – это мера неопределенности, чем она меньше, тем более предсказуемой, ожидаемой была последовательность, тем лучше модель смогла описать её. При значениях от 7 до 9 значение энтропии минимально, при такой длине контекста модель обучается лучше всего. Это позволяет оценивать среднюю длину контекста, в рамках которого можно предсказывать каждый следующий вызов процесса, а значит контролировать его исполнение.

В данном эксперименте участвовали только достаточно длинные потоки процесса.

Вывод

В данной работе был проведен анализ трасс системных вызовов процесса iexplore.exe. Для установления влияния контекста на совершаемые вызовы было предложено вычислять частную энтропию по алгоритму PPM. Было установлено, что алгоритм адаптируется к потоку вызовов самого длинного потока. При этом длина контекста, при котором модель PPM наиболее точно описывает последовательность, от 7 до 9 символов.

Литература

1. Vijay Janapa Reddi, Alex Settle, Daniel A. Connors, and Robert S. Cohn. 2004. PIN: a binary instrumentation tool for computer architecture research and education. In Proceedings of the 2004 workshop on Computer

architecture education: held in conjunction with the 31st International Symposium on Computer Architecture (WCAE '04). ACM, New York, NY, USA, , Article 22 . DOI=http://dx.doi.org/10.1145/1275571.1275600

2. Holy Father, "Hooking Windows API—Technics of Hooking API Functions on Windows," CodeBreakers J., vol. 1, no. 2, 2004;
3. Одеров Р.С., Тенсин. Е. (2011). Способы размещения своего кода в ядре ОС Microsoft Windows Server 2008. *Сборник трудов межвузовской научно-практической конференции «Актуальные проблемы организации и технологии защиты информации»*, 100-102. СПб.
4. J.G. Cleary, and I.H. Witten, Data compression using adaptive coding and partial string matching, IEEE Transactions on Communications, Vol. 32 (4), pp. 396–402, April 1984.