АНАЛИЗ ПОТОКОВ СИСТЕМНЫХ ВЫЗОВОВ ОС АНДРОИД ДЛЯ ПОИСКА ВРЕДОНОСНЫХ АКТИВНОСТЕЙ

Ханов А. Р., ст. преп. кафедры системного программирования СПбГУ.
Черепанов А. О., студент кафедры системного программирования СПбГУ,
mrneumann888@gmail.com.

Аннотация

В данной работе описан алгоритм сбора и анализа данных из потоков приложений в операционной системе Android непосредственно на самом низком уровне – на уровне системных вызовов к ядру Linux. Рассмотрены механизмы и технологии трассировки ядра, выделен оптимальный алгоритм для решения данной задачи. Тесты производительности показали, что предложенный в работе алгоритм сбора потоков системных вызовов несущественно замедляет работу мобильного устройства. Анализ полученных трасс позволяет получить первичное описание поведения приложений, а также предоставляет удобный фреймворк для более глубокого анализа.

Введение

С развитием технологий смартфоны на базе ОС Android завоевали повсеместную популярность. В день происходит около 1,5 миллиона активаций устройств Android и миллиарды установок приложений с официального магазина Google Play. Android стала одной из самых используемых операционных систем для смартфонов и планшетов. Ежегодно увеличивается их вычислительная мощность, добавляются новые датчики и функции, а также расширяются область использования, появляются встраиваемые версии ОС, развивается интернет вещей. Это может быть использовано разработчиками приложений для улучшения взаимодействия с пользователем, но с другой стороны, расширенные возможности, предоставляемые ОС Android для большей гибкости, также привлекают внимание авторов вредоносных программ, которые сместили акцент с ПК и начали создавать вредоносные приложения, предназначенные для смартфонов.

ОС Android позволяет приложениям создавать, изменять и удалять файлы, хранящиеся в системе, в том числе и файлы, принадлежащие другим приложениям. ОС Android даёт возможность управлять телефонными вызовами, что позволяет без ведома пользователя звонить на

премиум номера, похищая тем самым деньги с баланса SIM-карты, а также позволяет принимать, отправлять и удалять сообщения, что является потенциальной уязвимостью при использовании сервисов оповещений от банка, более того, существует возможность получения удалённого доступа к мобильному устройству и его функционалу вплоть до снимков экрана, камеры, записи видео и звука. Тем самым злоумышленники могут похитить личную информацию, деньги с лицевых счетов пользователя, а также причинить вред мобильному устройству, данным и приложениям, находящимся на нём.

Большинство угроз для Android приходят из сторонних приложений, скачанных не из официального магазина, в котором существуют алгоритмы Play Protect для проверки публикуемого в Play Market приложения на наличие вредоносного кода. Сторонние приложения не имеют надлежащих механизмов для защиты от вредоносных активностей. И обнаружить их могут разве что установленное на мобильном устройстве антивирусное ПО. Однако большинство антивирусных сканеров являются сигнатурными, это означает, что данное антивирусное ПО может обнаружить только известные вирусы. У антивирусных мониторов, сканирующих сетевые пакеты и файлы системы, изменяемые приложениями в реальном времени имеют высокий процент ложных срабатываний, а также сильно нагружают систему.

Цель работы

Цель данной работы заключается в разработке специализированного ПО, нацеленного на мониторинг поведения Android приложений, сбор необходимой для анализа информации и выявление приложений, которые нарушают предопределенную политику безопасности. В процессе исследования предметной области были сформулированы следующие задачи:

1. Провести обзор существующих исследований в области мониторинга активности процессов на платформе Android.
2. Изучить архитектуру ядра Linux для ОС Android.
3. Освоить технологию получения трасс системных вызовов ядра Android.
4. Собрать и протестировать специализированное ядро Linux, позволяющее вести запись трасс системных вызовов.
5. Разработать конвейер программ для анализа последовательностей номеров системных вызовов из полученных трасс.

6. Полученным инстументом проанализировать собранные данные и выявить паттерны поведения вредоносных программ.

Технологии, используемые для сбора данных

Для сбора данных были использованы технологии Android Open Source Project[1]. AOSP был выбран, как источник всех исходных кодов, включая кросс-компиляторы, эмулятор, отладчик, а также исходный код ядра и образов мобильных устройств для тестирования.

После эмпирических методов исследования оптимальной платформой для тестирования был выбран Android emulator. Имеющееся в наличии физическое устройство Alcatel Pop4 оказалось непригодным, так как компания Alcatel не опубликовала исходный код видоизменённого ядра Linux, которое находится под защитой лицензии GNU General Public License, без которого сборка модуля ядра оказалась практически невозможной. Стоит также отметить, что разработчиками ПО была сформирована петиция с обращением опубликовать исходный код, но она была проигнорирована. Таким образом, выбор пал в сторону эмуляторов. Однако, большинство эмуляторов (например, Genymotion, BlueStacks) подразумевают высокоуровневое тестирование системы на уровне приложений, т.е. не предоставляют возможности видоизменять ядро, эмулируют архитектуру процессора x86, а также имеют закрытые репозитории с исходным кодом запускаемых устройств на платформе Android. Android emulator, взятый из репозиториев AOSP отвечает всем требуемым характеристикам, а именно:

1. Имеет гибкую настройку запуска (без графической оболочки, без сохранения снимков состояний при выключении, вывод сообщений ядра в окно терминала).
2. С помощью AVD (Android Device Manager, в комплекте Android Studio) можно создать телефон любой архитектуры процессора, версии api, графическим разрешением.
3. Эмулирует архитектуру реального устройства — arm.

Специализированным ядром для Android emulator является Goldfish kernel, идущий вместе с образом мобильного устройства, но существует также возможность скомпилировать собственное ядро и модули к нему из исходников. Ядро Goldfish kernel также было взято из репозиториев AOSP.

Для взаимодействия с эмулируемым устройством был использован adb(android debug bridge) из Android Studio.

Для компиляции был использован кросс-компилятор для архитектуры arm на хосте с архитектурой x86_64, который также был взят из репозиториев AOSP.

Для снятия трасс системных вызовов был выбран фреймворк ftrace[2], потому что данный механизм анализа и отладки процессов встроен в ядро Linux, то есть при обработке системного вызова проверяется флаг отладки и возвращается информация о вызове на уровне команд ассемблера, что позволяет минимизировать нагрузку на производительность системы. Также потому что данный трассировщик предоставляет удобный программный интерфейс для чтения полученной информации.

Тестирование

В процессе тестирования были собраны трассы системных вызовов более 30 вредоносных приложений, взятых из открытых источников[3] и более 10 системных. 12 вредоносных приложений принадлежали к семейству JSMSers и имели схожий принцип работы. Вредоносные программы данного семейства предлагали пользователю подписаться на обновления, но вместо этого подключали платную подписку без ведома пользователя. Они отправляли СМС, а затем динамически создавали обработчик входящих сообщений, который скрывал сообщения, приходящие от подписки. Стоит отметить, что из-за архитектуры приложений в данном случае невозможно обнаружить вредоносную активность статическим анализом. В процессе димамического анализа изучалась каждая точка входа в приложение, были созданы различные условия запуска и работы приложений. Вирусы, как и системные приложения запускались на одну минуту, в течение которой производилась имитация пользовательской активности в приложении. За это время вредоносные приложения совершали от 5 до 10 тысяч системных вызовов, а суммарно за это время было зафиксировано около 60 тысяч обращений к системе. Анализ статистики вызвов показал, что вредоносные программы одного семейства схожи – они обращались к одним и тем же функциям, однако скорость генерации вызовов оказалась различна. Интенсивность вызовов вредоносных приложений семейства JSMSers была от 75 до 110 системных вызова в секунду. В качестве первичного анализа был выбран алгоритм сравнения по n-граммам, описанный Стефани Форрест[4]. В качестве критерия схожести было решено искать пересечения биграмм и триграмм вызовов. Анализ биграмм и триграмм потоков системных вызовов приложений, принадлежащих к семейству JSMSers, показал совпадение в 54 и 115 кортежей соответственно. Аналогично были проанализированы более 10 безвредных и системных приложений. Из полученных кортежей n-грамм вредоносных и безобидных приложений

были составлены контрольные множетсва, которые использовались для определения степени опасности контрольных приложений, которые в анализе еще не учавствовали. Результаты работы показали(см. Граф. 1), что данная модель определяет вредоносные приложения с низкой степенью точности.

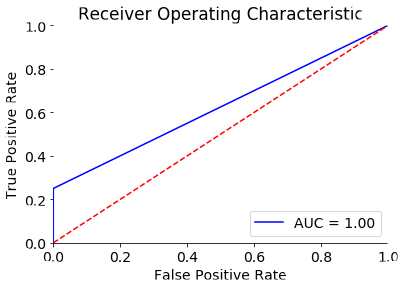

График 1: Оценка качества n-граммной модели

Заключение

Резкое увеличение использования устройств на платформе Android привело к созданию и распространению огромного количества вредоносных программ для Android. В этом тезисе был предложен фреймворк для сбора и анализа приложений Android с использованием методов динамического анализа. Эффективность предложенного фреймворка была подтверждена тестированием на специально собранном ядре Linux, позволяющим снимать трассы системных вызовов, где он показал несущественную нагрузку на производительность системы.

Литература

1. AOSP — 2018 — URL: https://source.android.com/
2. ftrace — 2008 — URL: https://www.kernel.org/doc/Documentation/trace/ftrace.txt
3. android-malware — 2019 — URL: https://github.com/ashishb/android-malware
4. Steven A. Hofmeyr, Stephanie Forrest, Anil Somayaji. Intrusion Detection using Sequences of System Calls — 1998 — URL: https://www.cs.unm.edu/~forrest/publications/jcs-sequences-of-system-calls-98.pdf