УДК 519.712.2

ВИРТУАЛЬНЫЕ МАШИНЫ КАК СРЕДСТВО ЗАЩИТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Лебедев А.Н., Курганов Н.С.

Московский государственный технический университет имени Н.Э.Баумана
Москва, Российская Федерация

Введение. Многие разработчики ПО прибегают к защите кода путем использование виртуальных машин (далее VM). Единые подходы к анализу структуры VM отсутствуют, что сильно затрудняет анализ программного обеспечения. В докладе анализируются общие подходы к анализу VM, например, метод анализа "black box", а также визуализация структуры работы VM и статистика по количеству приходящих на анализ файлов в вирусную лабораторию, защищенных VM.

Вне зависимости от сложности кода, который виртуализирован, за короткий промежуток времени вполне реально составить общую картину его работы. Знание типовой внутренней структуры VM и некоторые тонкости ее работы в сумме дают возможность получить больше информации, используя возможности отладчика. В некоторых случаях возможно обойтись скриптовым языком отладчиков и выполнить «трассировку кода» VM, не прибегая к написанию сложных инструментов.

Виртуальная машина – это некий процессор (то что эмулируется виртуальной машиной) внутри основного PE-файла (исполняемого файла под Windows) со своим набором инструкции (т.е. инструкций, отличных от ассемблерных стандартный инструкций). Наиболее популярные VM это VMProtect и Themida.

Используются в основном в защитных системы от кражи лицензионных видеоигр типа DENUVO, SecuROM, StarForce и т. д.

Общий алгоритм работы виртуальной машины:

1. В точку входа виртуальной машины передается шифрованный указатель на начало ленты P-code.
2. Инициализируются виртуальные регистры и расшифровывается указатель на начало ленты P-code, который был передан в VM ранее.
3. Загружается в одни из виртуальных регистров указать на таблицу хендлеров.
4. Происходит чтение и расшифровка ленты P-code.
5. Расшифрованный байт ленты P-code в большинстве случаев складывается с указателем на выполняемую инструкцию в таблице хендлеров. Таким образом из таблицы хендлеров получаем адрес следующей выполняемой виртуальной инструкции (хендлера).
6. Лента P-code считывается, пока не будет достигнут ее конец и выход из виртуальной машины.
7. Аналогично происходит выполнение инструкций виртуальной машины из таблицы хендлеров.
8. По окончанию работы виртуальный машины достигается выход из виртуальной машины и происходит дальнейшее выполнение программы.
9. **Sandbox**, она же песочница. Это некая среда, которая на 100% эмулирует любую операционную систему (далее ОС) на компьютере. Работает аналогично средам виртуализации таким, как VMware, Virtual box, но у нее отсутствуют артефакты сред виртуализации.

Рисунок 1 – Структура виртуальной машины

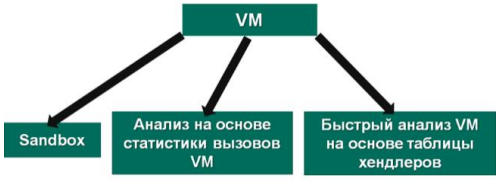

Рисунок 2 – Подходы к анализу VM

Анализ на основе статистики вызовов VM – метод анализа основан на статистики по тому, насколько частно VM внутри себя вызывает те или иные функции. Благодаря этому можно сосредоточить анализ именно на тех функция, которые вызываются редко. Таким образом данный метод позволяет анализировать не всю виртуальную машину, а только действительно важные функции, которые использует VM.

В подавляющем большинстве, особенно если это языковые операторы memcpy и WinAPI, ленту P-code (в случае виртуальной машины VMProtect она находится в push) можно сразу начать анализ именно с этой процедурой. Для чего это нужно? Мы можем просканировать весь файл и посчитать, сколько всего push (p-code) call VM_ENTER инструкций в нем. Таким образом мы можем сконцентрироваться на анализе функций, которые вызываются редко, и они отвечают за всю логику работы VM.

Быстрый анализ VM на основе таблицы хендлеров – это непосредственный разбор самих

функции – команды, которые поддерживает данная VM. Данные команды содержатся в таблице хендлеров. Зная описания каждой команды в таблице хендлеров мы знаем как работает VM.

Если VM использует таблицу хендлеров, то её правильное декодирование поможет сразу, в статическом режиме, получить хендлеры VM, а в дальнейшем – поставить точки останова в ней, что предоставит возможность снятия трассировок. Это крайне важный момент, который обеспечит быстрый анализ VM на низком уровне. Если удастся собрать трассировку выполнения VM, то можно построить некоторую модель поведения, которую можно использовать в качестве детектирования для вредоносной программы. Что, собственно, представляет собой таблица хендлеров и что понимается под словом хендлер?

VM – в первую очередь, это процессор, который имеет свой ограниченный набор инструкций, которые мы и называем хендлером: прочитать, записать, сложить и т.д. В подавляющим большинстве случаев этот размер = 256 (0-0xFF, 1 байт) [2]. Это размер по умолчанию для таблицы хендлеров. Процессор должен знать, точный адрес, где находится каждая инструкция, чтобы её использовать.

Любой исполняемый файл, защищенный VM, может рассматриваться вирусным аналитиком или исследователем не только, как упакованный разными упаковщиками исполняемых файлов, например, UPX. Из него получается следующую информация:

– общее количество VM, которые содержаться в файле;

– таблица хендлеров для каждой машины. Типы хендлеров. Характерная модель поведения VM, присущая именно этой программе;

– на основании таблицы хендлеров составляется процент покрытия кодом VM в соотношении с основным незашифрованным кодом программы.

Как минимум, эти метрики и данные используются в качестве детектирования похожих вредоносных файлов. А собственно, как максимум – возможность девиртуализации кода, защищенного VM.

В итоге имеем общую схему внутреннего устройства виртуальной машины (см. Рис. 3).

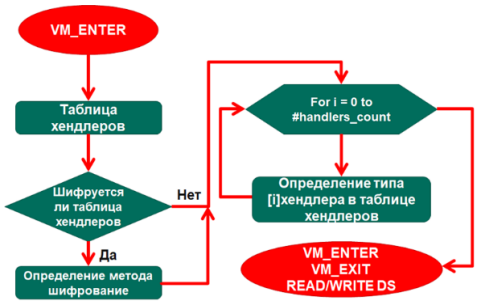

Рисунок 3 – Внутреннее устройство виртуальной машины

Таким образом, виртуальные машины могут рассматриваться как одно из эффективных и удобных средств защиты программного обеспечения от злонамеренных воздействий со стороны как вирусного ПО, так и других хакерских приемов.

Литература

1. Крепыш, 2011. Хабрахабр. https://habr.com/ru/sandbox/26302/.

2. 2019. VMProtect – Справочник исследователя программ. https://exelab.ru/faq/VMProtect.

3. eXeL@B, 2019 Реверс VMProtect. https://exelab.ru/f/index.php?action=vthread&forum=13&topic= 25221.

УДК 519.712.2

О ПРОБЛЕМЕ РЕАЛИЗАЦИИ ДИСПЕТЧЕРА ДОСТУПА К ДОКУМЕНТАМ

Лебедев Г.А., Родионов Д.Е., Лебедев А.Н.

Московский государственный технический университет имени Н.Э.Баумана
Москва, Российская Федерация

В любой автоматизированной системе в защищенном исполнении (АСЗИ) механизм управления доступом субъектов к объектам доступа выполняет основную роль в обеспечении конфиденциальности защищаемых данных. Реализация этого механизма обычно строится на концепции единого диспетчера доступа (ДД). Сущность этой концепции состоит в том, что диспетчер доступа выступает посредником-контролером при всех обращениях всех субъектов системы к ее объектам.

В качестве базовой платформы для использования в АСЗИ выступает операционная система Astra Linux Special Edition. Данная операционная система серифицированна в Российской Федерации для обработки информации ограниченного доступа, различных уровней конфиденциальности [1].

При разработке АСЗИ возникает задача реализации механизма разграничения доступа к объектам (документам), доступ к которым не может быть разграничен непосредственно ОС, так как в ОС объектами разграничения длоступа являются, в первую очередь, файлы. В этом случае возникает задача разработки защищенного приложения – ДД для реализации данных функ-