

Н.И.Фурсова

ИНТРОСПЕКЦИЯ ВИРТУАЛЬНЫХ МАШИН

В работе описан метод мониторинга программных систем и приложений в виртуальной машине. Проект представляет собой фреймворк, основанный на плагинах, получающих информацию из работающей системы. Фреймворк может быть использован для отладки, динамического анализа, обнаружения вторжений и т.д.

Ключевые слова: Инструментирование, динамический анализ, виртуальная машина, интроспекция

Введение

В работе рассматривается вопрос отладки и анализа программных систем и приложений, работающих в виртуальной машине QEMU [1]. В качестве средства обеспечения отладки и анализа выбран метод интроспекции.

Под интроспекцией понимается процесс получения высокоуровневой информации из работающей системы [2]. Полносистемные симуляторы, к которым относится и QEMU, спроектированы так, что позволяют получать доступ к каждой части гостевого состояния, например, значениям регистров и содержимому памяти. Но этого недостаточно, так как данных о большей части содержащихся данных, таких как структуры, определенные в гостевой операционной системе или программах, не предоставлено. Получить можно только ту информацию, которая необходима процессору для выполнения команд, однако этого мало чтобы проводить высокоуровневый анализ. Отсюда следует, что для того, чтобы понять состояние компьютерной системы, необходимо получить знания о структурах, с которыми она работает.

Описание метода

Предлагается система для интроспекции виртуальных машин, структура которой представлена на рис. 1. Работа виртуальной машины разделена на пять уровней, на каждом из которых выделено несколько сущностей, которые будут подвергаться анализу.

Уровень исходных кодов	Отладочная информация			
Уровень приложения	Модули		Приложения	
Уровень ОС	Файлы		Потоки и процессы	
Уровень низкий	Виртуальная память		Контекст исполнения	Системные вызовы
Уровень аппаратный	Трансляция	ЦП	Блок управления памятью	Периферия

Рис. 1. Структура системы для интроспекции

Система должна иметь модульное строение, где каждый модуль — это динамическая библиотека. Модули наделены возможностью взаимодействовать друг с другом, например, выходные данные одного модуля могут служить входными данными для другого.

Такая архитектура максимально отвечает требованиям, поскольку важно, чтобы модули были независимы от виртуальной машины. Это связано с сопровождением кода, разработкой новых модулей, добавлением новых архитектур и гостевых операционных систем.

Реализация

В рамках проделанной работы был реализован модуль «Системные вызовы» и затронут модуль «Файлы». Перехватываются такие функции, как создание файла (NtCreateFile, creat для Windows и Linux соответственно), открытие файла (NtOpenFile, open), чтение (NtReadFile, read), запись (NtWriteFile, write) и закрытие файла (NtClose, close).

Общая схема работы модулей представлена на рис. 2.

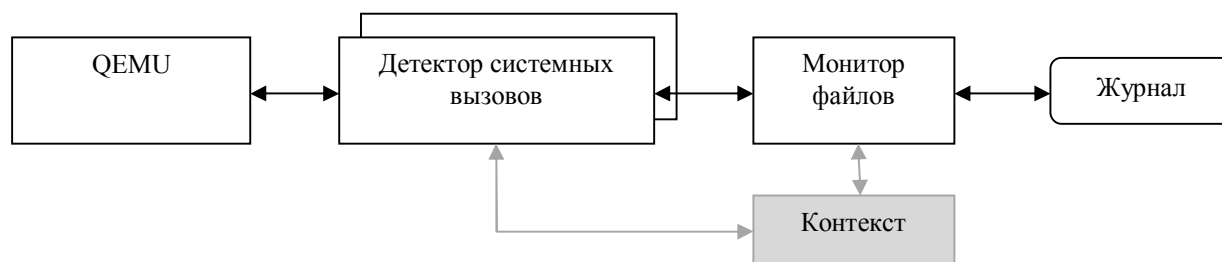


Рис. 2. Отслеживание системных вызовов

Детектор системных вызовов выполняет функцию перехвата и фильтрации происходящих системных вызовов, которые затем переправляются в монитор файлов, осуществляющий их обработку и записывающий журнал. Модуль контекста отвечает за определение текущего потока выполнения.

Системные вызовы выполняются в режиме ядра и наделены идентификаторами [3-4], используемыми для вызова соответствующих подпрограмм.

Для перехода в привилегированный режим в операционных системах существуют специальные инструкции: в Windows это sysenter, а в Linux - int 80h. Именно эти инструкции отслеживаются по их кодам во время работы эмулятора. Идентификаторы системных вызовов хранятся в регистре eax, прочитав который, вызывается соответствующая функция плагина, отвечающая за запись системного вызова в журнал. В него попадает информация о параметрах вызываемых функций. Для определения возвращаемых значений необходимо перехватывать завершение системных вызовов.

Инструкциями завершения вызовов являются sysexit (Windows) и iret (Linux), которые так же перехватываются в процессе выполнения эмулятора. После того, как системный вызов завершен, появляется возможность получить такие данные, как, например, описатель открытого/созданного файла или буфер, который был прочитан из файла.

Заключение

В статье описан подход для создания фреймворка для динамического анализа. Фреймворк будет состоять из большого количества взаимодействующих плагинов.

На настоящий момент реализован основополагающий плагин, отслеживающий системные вызовы. В результате его работы получается журнал системных вызовов, содержащий информацию о параметрах и возвращаемых значениях функций-обработчиков вызовов.

На рисунках 3 и 4 представлены фрагменты журнала системных вызовов для операционных систем Windows и Linux соответственно.

```

NtOpenFile (pFileHandle 0x15f69c, DesiredAccess 0x12019f, pObjectAttributes 0x15f684 {Length
0x18, RootDirectory 0x0, ObjectName 0x15f67c {Length 0x30, MaximumLength 0x32, pBuffer 0x15f6a0
{name: \SystemRoot\bootstat.dat}}, Attributes 0xc0, SecurityDescriptor 0x0, SecurityQualityOfService
0x0}, IoStatusBlock 0x15f674, ShareAccess 0x0, OpenOptions 0x20)
File handle = 0x1c
NtReadFile (FileHandle 0x1c, Event 0x0, ApcRoutine 0x0, ApcContext 0x0, AIoStatusBlock 0x15f87c,
pBuffer 0x15f88c, Length 0x4, ByteOffset 0x15f884, Key 0x0)
NtClose (FileHandle 0x1c)
    
```

Рис. 3. Фрагмент журнала для Windows

```

sys_open (pfilename 0x805533d {name: /etc/fstab}, access 0x8000, permission 0x0}
File handle = 0x4
sys_read (handle 0x4, pBuffer 0xb7f7d000, nbyte 0x400)
sys_close (handle 0x4)
    
```

Рис. 4. Фрагмент журнала для Linux

1. QEMU [Электр. ресурс]. URL: http://wiki.qemu.org/Main_Page (дата обращения: 07.05.2015).
2. Virtual machine introspection [Электр. ресурс]. URL: <http://www.securityweek.com/vm-introspection-know-your-virtual-environment-inside-and-out> (дата обращения: 07.05.2015).
3. Windows system call table [Электр. ресурс]. URL: <http://j00ru.vexillium.org/ntapi/> (дата обращения: 07.05.2015).
4. Linux system call table [Электр. ресурс]. URL: http://docs.cs.up.ac.za/programming/asm/derick_tut/syscalls.html (дата обращения: 07.05.2015).

References

1. QEMU [Elektr. resurs]. URL: http://wiki.qemu.org/Main_Page (data obrashcheniya: 07.05.2015).
2. Virtual machine introspection [Elektr. resurs]. URL: <http://www.securityweek.com/vm-introspection-know-your-virtual-environment-inside-and-out> (data obrashcheniya: 07.05.2015).
3. Windows system call table [Elektr. resurs]. URL: <http://j00ru.vexillum.org/ntapi/> (data obrashcheniya: 07.05.2015).
4. Linux system call table [Elektr. resurs]. URL: http://docs.cs.up.ac.za/programming/asm/derick_tut/syscalls.html (data obrashcheniya: 07.05.2015).

Fursova N.I. Introspection of virtual machines. The paper describes a method for monitoring the systems and applications in a virtual machine. The project is a framework based on plugins, receiving information from the running system. The framework can be used for debugging, dynamic analysis, intrusion detection, etc.

Keywords: Software instrumentation, dynamic analysis, virtual machine, introspection

Сведения об авторе. Н.И.Фурсова — аспирант, младший научный сотрудник; научно-исследовательский центр НовГУ; e-mail: natalia.fursova@ispras.ru

Статья публикуется впервые. Поступила в редакцию 10.09.2015.