В. Л. Евсеев, В. А. Данилкин, С. В. Рогачев, А. И. Трибунский

ГИПЕРВИЗОРЫ БЕЗОПАСНОСТИ, ИСПОЛЬЗУЮЩИЕ ТЕХНОЛОГИЮ АППАРАТНОЙ ВИРТУАЛИЗАЦИИ

В последние пять лет одним из наиболее актуальных направлений в области защиты информации от несанкционированного доступа (НСД) стала идея гипервизоров безопасности, использующих технологию аппаратной виртуализации, в частности технологию виртуализации Intel/AMD для x86 процессоров [1].

Характерной чертой современных систем является то, что они не поддерживают технологию виртуализации на аппаратном уровне, а концепция паравиртуализации также оказывается частично или полностью не применимой для них [2], поскольку открытая модификация гостевого программного обеспечения, исполняемого на данной платформе, может быть недоступной. Применение чистой эмуляции для организации защищенных сред малоэффективно в плане быстродействия гостевого программного обеспечения. В то же время часто отсутствует необходимость виртуализации всей аппаратной платформы, но требуется защита программной среды на уровне процессов [3], функционирующих в системе. Например, для организации многозадачных сред на аппаратуре, не поддерживающей механизмы защиты памяти.

В рамках данной работы рассматривается новая концепция программных гипервизоров безопасности, способных в ряде случаев решить обозначенные проблемы.

Анализ модели защиты, реализуемой гипервизором безопасности для платформ с аппаратной поддержкой технологии виртуализации

Рассмотрим базовые принципы, заложенные в решения, построенные на технологии аппаратной виртуализации [1].

Аппаратно-программное виртуализационное решение (гипервизор) представляет собой комплекс программного обеспечения, осуществляющий начальную низкоуровневую инициализацию аппаратных средств с целью создания независимого контекста исполнения, дублирующего в определенной мере аппаратную среду реальной машины.

Такой контекст формируется набором регистров виртуальной машины, к которым относятся регистры общего назначения, сегментные регистры, регистры управления режимами процессора, MSR-регистры; набором параметров, управляющих процессом исполнения виртуальной машины, и некоторыми другими значениями. Сформированный контекст сохраняется в специальной структуре базы данных, а управление им контролируется со стороны аппаратного обеспечения.

Центральный элемент гипервизора, его ядро, является циклом, в котором последовательно происходят операции входа в гостевую систему, выхода из нее по некоторой причине и обработки события, послужившего причиной выхода на уровень гипервизора из гостевой системы.

Идея гипервизора безопасности основана на внедрении таких обработчиков событий (выходов), которые могут обеспечить обнаружение и устранение вредоносной активности, происходящей внутри гостевой системы.

Важно заметить, что операции входов и выходов выполняются аппаратным обеспечением: встречая соответствующие инструкции в коде гостевой системы, процессор прерывает исполнение гостевого контекста, производя передачу управления гипервизору. Это позволяет избежать модификации гостевого программного обеспечения при сохранении высокого уровня отзывчивости всей системы за счет исполнения гостевого кода на реальном процессоре.

Не следует воспринимать гипервизоры безопасности исключительно как средства полноценной виртуализации. Конечно, в качестве гостевого программного обеспечения часто выступают целые

операционные системы, но существуют и концепции по организации защищенных аппаратных контекстов на уровне отдельных процессов ОС. Таким образом, технология аппаратной виртуализации может быть применена для создания операционных систем нового типа. Каждый процесс в таких ОС исполняется в собственной защищенной среде, при этом события безопасности обрабатываются общим для всех процессов программным комплексом — гипервизором уровня приложений.

Принципы построения систем виртуализации без ее аппаратной поддержки

В случае, если платформа не поддерживает технологию аппаратной виртуализации, гипервизор строится на основе эмулятора или концепции паравиртуализации.

В случае чистой эмуляции процессор и аппаратное обеспечение виртуальной машины представляются программными эквивалентами — их моделями. Основной цикл эмуляции может рассматриваться в виде программы последовательного считывания и интерпретации инструкций гостевой системы.

Часто используется оптимизация на основе подхода, известного как бинарная трансляция. Такой подход позволяет сократить время, затрачиваемое на интерпретацию инструкций, за счет трансляции блоков инструкций в код реальной машины и его последующего исполнения.

Обработчики событий безопасности внедряются в такую виртуальную машину и функционируют подобным аппаратному гипервизору образом.

Данный подход неприменим, если речь идет об аппаратуре с низкой производительностью, поскольку отзывчивость гостевой системы в таком случае будет чрезвычайно низкой.

Но он зарекомендовал себя при создании сэндбоксов в антивирусных приложениях, необходимых для реализации проактивных технологий антивирусной защиты [4] на основе динамического анализа программного обеспечения во время его исполнения.

С целью организации высокопроизводительных виртуализационных средств применяется подход паравиртуализации, подразумевающий открытую модификацию гостевого программного обеспечения. За счет подобной модификации программное обеспечение, исполняемое в гостевой системе, осуществляет доступ к аппаратуре через применение операции гипервызова. Гипервызов, являющийся аналогом системного вызова в операционных системах, осуществляет делегирование обязательств кода гостевой системы гипервизору. Таким образом, гипервизор выполняет часть работы за код гостевой системы. Иногда применяются различные аппаратные технологии для еще большего упрощения модели взаимодействия гипервизора и гостевой системы, к таким средствам можно отнести, например, возможности, предоставляемые защищенным режимом процессоров, построенных на архитектуре x86.

Обработчики безопасности в данном случае могут быть встроенными на уровне кода, обрабатывающего события гипервызовов.

Обычно такие решения осуществляют полную виртуализацию машины и используются для запуска модифицированных операционных систем. Данный подход не нашел широкого применения в области организации защищенных сред для исполнения отдельных прикладных процессов.

В некоторых случаях открытая модификация гостевого программного обеспечения невозможна. Например, в большинстве информационных систем военного назначения отсутствуют открытые исходные коды, недопустимо свободное расширение программного обеспечения. Из соображений производительности неприменимы обычно и системы на основе эмуляторов.

Именно для такого класса систем и предлагается использовать рассматриваемый подход.

Модель защиты от НСД на базе программного гипервизора безопасности для исполнения программного обеспечения, не модифицируемого штатными средствами

Предлагаемая модель в простейшем приближении основывается на идеях модификации бинарного кода гостевого программного обеспечения, проверки соответствия данного кода ряду

ограничений в момент его загрузки гипервизором для исполнения и контроля исполнения кода за счет обработки вызовов, размещенных в бинарном коде в момент его модификации. Таким образом, планируется добиться большей управляемости машинного кода.

Важно заметить, что данная модель может быть применена не для всех типов гостевого программного обеспечения. В статье не рассматриваются вопросы виртуализации целых операционных систем, поскольку это может оказаться просто невозможным в рамках данной модели. Предлагается использовать ее для организации безопасных операционных сред, исполняющих в защищенных контейнерах код программ неизвестного происхождения.

Рассмотрим более подробно каждый компонент данной системы.

Подсистема модификации бинарного кода представляет собой программное обеспечение, осуществляющее поиск «опасных», с точки зрения данного программного комплекса, инструкций и модифицирующее код путем добавления операций, отчасти похожих на гипервызовы, применяемые в средствах паравиртуализации. Каждый такой вызов, исходя из семантики инструкции, находящейся после него, осуществляет запрос разрешения на ее исполнение у гипервизора. В процессе модификации бинарного кода рассматриваемая подсистема осуществляет исправление указателей и требуемую релокацию отдельных блоков, таким образом сохраняя все внутренние переходы и обращения к данным невредимыми. Это необходимо, поскольку добавление нового кода нарушает адресацию внутри исполняемого модуля.

Подобная модификация значительно упрощается, если бинарный код организован в виде объектного модуля, в противном случае время модификации значительно увеличивается. Тем не менее время модификации не является критичным, поскольку эта операция осуществляется лишь один раз на стадии подготовки программного обеспечения к запуску под управлением гипервизора такого типа.

Следует выделить класс инструкций, который требует контроля со стороны гипервизора и, соответственно, предварения их гипервызовами с запросом разрешения на исполнение этих инструкций. Среди таких инструкций операции перехода и управления ходом исполнения программы, операции работы со стеком, операции обращения к памяти на запись и чтение. В зависимости от дисциплины управления безопасностью данный набор может значительно расшириться.

Подсистема проверки соответствия осуществляет загрузку уже модифицированного исполняемого модуля программы, при этом система производит поиск «опасных» инструкций по сигнатурам и контролирует наличие перед ними предваряющих вызовов. В случае если часть инструкций осталась без изменений, модуль помечается как недоверенный и не загружается.

Собственно гипервизор выполняет обработку запросов от исполняемого программного обеспечения. При поступлении запроса гипервизор проводит проверку, основанную на распознавании семантики инструкции. В случае если данная инструкция может быть выполнена, гипервизор возвращает управление коду программы и он продолжает свое исполнение.

В качестве примера рассмотрим запрос от программы на запись в определенную область памяти. Программа с помощью инструкции «mov» записывает по некоторому адресу, который может быть взят из кода инструкции, целое число. Гипервизор получает управление непосредственно перед исполнением данной инструкции, поскольку до того была выполнена модификация бинарного кода данной программы соответствующим компонентом. Обработчик безопасности проверяет, принадлежит ли адрес, по которому производится запись, данной программе или нет. На основе проведенного анализа принимается решение о дальнейшем исполнении или остановке программы.

Аналогичные проверки могут быть выполнены для других инструкций, предваренных соответствующим вызовом гипервизора.

Следует отметить, что данный подход имеет общие черты с тем, что применяется в гипервизорах, основанных на технологии аппаратной виртуализации. Действительно, в гостевое

программное обеспечение на стадии разработки не закладывается функционал взаимодействия с гипервизором, не происходит модификации его исходного кода, встраивания дополнительных модулей, но, поскольку бинарный код модифицируется специальным программным обеспечением перед исполнением, выходы на уровень гипервизора осуществляются прозрачно, как будто это происходит на аппаратном уровне. Аналогична и концептуальная модель функционирования такого гипервизора на уровне цикла входов, выходов и обработки событий.

Выводы

Рассмотренный подход может быть применен для организации безопасных встраиваемых операционных сред, позволяющих исполнять стороннее программное обеспечение, поставляемое в виде бинарных образов. Кроме того, данный подход применим и в случае отсутствия любых аппаратных средств защиты памяти на целевой платформе, так как он позволяет исключить доступ к коду и данным иных процессов, равно как и самомодификацию кода приложений. Особенно это актуально для целевых встраиваемых систем военного назначения.

Возможна адаптация существующих операционных систем с целью выполнения их процессов в подобных защищенных средах.

К недостаткам данного подхода можно отнести увеличение объема бинарного кода после модификации в среднем на 30–40 %, а также саму необходимость, пусть и однократной, его модификации, которая требует временных затрат.

СПИСОК ЛИТЕРАТУРЫ:

1. *Shinagawa T. and etc*. BitVisor: A Thin Hypervisor for Enforcing I/O Device Security // Proc. 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2009). Washington, DC, USA. March 11–13, 2009. С. 121–130.

2. *Бетелин В. Б., Галатенко В. А., Годунов А. Н., Грюнталь А. И*. Обеспечение информационной безопасности систем на программной платформе ОС Windows 2000 // Московский университет и развитие криптографии в России. Материалы конференции в МГУ 17–18 октября 2002 г. М.: МЦНМО, 2003. С. 254–266.

3. *Зегжда Д. П., Вовк А. М*. Защищенная гибридная операционная система «Linux over Феникс» // Математика и безопасность информационных технологий. Материалы конференции в МГУ 28–29 октября 2004 г. М.: МЦНМО, 2005. С. 91–117.

4. *Казарин О. В., Скиба Л. М*. Парадигма проактивной безопасности компьютерных систем // Защита информации. INSIDE. 2009. № 6. С. 2–7.