ЗАЩИТА КОНФИДЕНЦИАЛЬНОСТИ ПРИЛОЖЕНИЙ, РАБОТАЮЩИХ ПОД УПРАВЛЕНИЕМ НЕДОВЕРЕННОЙ ОПЕРАЦИОННОЙ СИСТЕМЫ

Ефремов Денис Валентинович
Студент
МГУ им. Ломоносова
Тел. 89104821413
e-mail: yefremov.denis@gmail.com

Введение

В статье излагается общая схема функционирования программной системы, разрабатываемой в ИСП РАН. Целью рассматриваемой системы является обеспечение защиты конфиденциальности данных на компьютере, подключенном к публичной сети (Интернет) и управляемом недоверенной операционной системой (на примере GNU/Linux). Предполагается, что операционная система (ОС) может содержать «закладки», предоставляющие вредоносному коду неограниченный доступ ко всем программным и аппаратным ресурсам персонального компьютера в обход штатных систем защиты.

Ряду доверенных приложений для нормального функционирования требуется доступ к публичной сети. Это сетевое соединение при отсутствии должного контроля может быть использовано вредоносным кодом в ядре ОС для организации утечки конфиденциальных данных. Задача системы безопасности – предотвратить утечку данных.

В предположении наличия вредоносного кода в ядре ОС, надежным способом предотвращения утечки данных является физическая изоляция компьютера от сетевого соединения, однако, тогда пострадают все легальные приложения, которым необходим доступ к сети. Предлагаемое решение основано на изоляции ОС, но при этом сохраняет работоспособность заданных доверенных приложений прозрачным для них образом, предоставляя им и только им доступ к сетевым ресурсам.

Система защиты построена на использовании технологии аппаратной виртуализации [1]. Эта технология широко используется для решения различного рода задач компьютерной безопасности. Среди них исследование поведения вредоносного кода [2], повышение надежности ядра операционной системы путем размещения драйверов периферийных устройств в отдельных виртуальных машинах [3],

организация защищенной среды выполнения для пользовательских программ [4, 5].

Технология виртуализации позволяет выполнять ОС в аппаратной виртуальной машине (ВМ) под управлением сравнительно небольшой по размеру системной программы – монитора виртуальных машин (гипервизора) [6]. ВМ представляет собой эффективный изолированный дубликат реальной машины. Выполнение в ней ОС не требует внесения каких-либо изменений в ее код. Гипервизор, в зависимости от сложности его реализации, может поддерживать одновременное функционирования нескольких ВМ на одном персональном компьютере и либо выделять каждой из них в монопольное использование некоторое подмножество физических ресурсов, либо обеспечивать совместный доступ нескольких ВМ к одному и тому же ресурсу [7].

Характерной особенностью систем защиты, направленных на предотвращение угроз нарушения конфиденциальности является то, что подобная система с одной стороны должна полностью контролировать весь поток информации, передаваемый с компьютера через сетевой интерфейс, а с другой стороны быть надежно защищенной от любых вредоносных воздействий со стороны привилегированного кода, выполняющегося на уровне ядра ОС.

Функционирование гипервизора на более высоком аппаратном уровне привилегий позволяет полностью контролировать выполнение, как кода ОС, так и пользовательских программ, оставаясь при этом аппаратно защищенным от вредоносного воздействия со стороны кода в ВМ, в том числе, кода, выполняющегося в привилегированном режиме.

Одним из важных требований, предъявляемых к гипервизору, на основе которого строится система безопасности, является компактность его размера [8]. Это требование объясняется тем, что гипервизор имеет неограниченного доступ к физическим ресурсам вычислительной системы, поэтому от надежности его кода (отсутствия в нем уязвимостей) зависит защищенность всей системы в целом.

Архитектура

В рассматриваемой системе гипервизор обеспечивает одновременное выполнение двух, полностью изолированных друг от друга, виртуальных машин – вычислительной (основной) и коммуникационной (сервисной). Обе виртуальных машины работают под управлением одинаковой, недоверенной операционной системой.

Пользователь работает в вычислительной ВМ, которая управляет всеми устройствами за исключением сетевого адаптера, через который возможна утечка информации. Вычислительной ВМ предоставляется строго один процессор (ядро процессора). В ней расположены

критически важные данные и выполняются программы (как доверенные, так и недоверенные), обрабатывающие эти данные. Данные хранятся в открытом (нешифрованном) виде, и система защиты не ограничивает доступ процессов (как системных, так и пользовательских) к этим данным, что позволяет обрабатывать их произвольными программами, в т.ч. недоверенными.

Во время запуска вычислительной виртуальной машины гипервизор блокирует ей доступ к сетевому интерфейсу. Операционная система, выполняющаяся в ней, считает, что сетевой адаптер физически отсутствует. Поэтому любая попытка установить сетевое соединение изнутри этой виртуальной машины и передать данные на удаленный компьютер неминуемо приведет к ошибке. Таким образом, вредоносный код, выполняющийся на любом уровне аппаратных привилегий процессора внутри вычислительной виртуальной машины, даже если ему удалось получить доступ к критически важным данным, не сможет передать их за ее пределы.

Среди всех пользовательских программ, расположенных внутри вычислительной виртуальной машины, выделен ряд доверенных программ, которым для корректного функционирования необходимо взаимодействовать с удаленными компьютерами в сети. Для предоставления им сетевого доступа используется коммуникационная виртуальная машина. Эта ВМ выполняется в фоновом режиме. Необходимо отметить, что единственное периферийное устройство, обслуживаемое коммуникационной ВМ – это сетевой адаптер. Любая программа в этой виртуальной машине может взаимодействовать с удаленными компьютерами в сети. В силу изоляции виртуальных машин, обеспечиваемой гипервизором, программное обеспечение в коммуникационной виртуальной машине не может получить доступ к данным, расположенным внутри вычислительной виртуальной машины.

Драйвер сетевой карты и TCP/IP стек могли бы быть реализованы в гипервизоре. Использование дополнительной виртуальной машины вносит накладные расходы на процессорное время и машинную память. Однако требование минимальности доверенного кода является приоритетным. Так же это позволяет поддерживать модульность всей системы защиты. Вполне возможно использование новых виртуальных машин для обслуживания доступа к иным периферийным устройствам, например, к жёсткому диску.

В рассматриваемой системе безопасности коммуникационная ВМ требуется только на время работы доверенных приложений. В остальные периоды времени ОС вычислительной ВМ может непосредственно работать на компьютере, задействуя все его ресурсы в

полном объеме (все ядра процессора), в т.ч. сетевой адаптер, если только пользователь гарантирует отсоединение сетевого кабеля от адаптера в эти интервалы времени.

Поддержка сетевого взаимодействия для доверенных процессов реализуется посредством удаленного обслуживания необходимого набора системных вызовов в сервисной ВМ. Гипервизор перехватывает системные вызовы, поступающие от доверенных процессов, анализирует их и, при необходимости, передает на обслуживание в коммуникационную ВМ. Системные вызовы других процессов, а также остальные системные вызовы доверенных процессов, обслуживаются локально в вычислительной ВМ. Выполнение процесса на время удаленного обслуживания системного вызова приостанавливается. Остальные процессы в вычислительной ВМ при этом продолжают выполняться.

Единственная информация, передаваемая за пределы вычислительной ВМ, – это явным образом специфицированные доверенным процессом значения параметров системных вызовов, причем передача информации за пределы ВМ осуществляется доверенным кодом (гипервизором).

Доверенные процессы выполняются под управлением недоверенной операционной системы. В отсутствие должного контроля со стороны системы безопасности вредоносный код в ядре операционной системы может загрузить в адресное пространство доверенного процесса необходимый код, передать на него управление, и доверенный процесс от своего имени осуществит все необходимые действия по доставке критически важной информации на удаленный компьютер, контролируемый злоумышленником. Для предотвращения такого вредоносного воздействия система безопасности защищает контекст доверенного процесса от несанкционированной модификации со стороны любого кода в вычислительной ВМ, в т.ч. привилегированного.

Механизмы обмена данными между гипервизором и ВМ

Удаленное обслуживание системных вызовов реализуется гипервизором совместно с другими компонентами системы, функционирующими в обеих ВМ – вычислительной и сервисной. В ходе инициализации системы в ядро ОС в вычислительной и сервисной ВМ динамически загружаются модули.

Каждый модуль выделяет непрерывное пространство физической памяти для организации кольцевого буфера, регистрирует несколько обработчиков прерываний, при помощи которых гипервизор извещает виртуальную машину о событиях, требующих обработки и сообщает эту информацию (адрес буфера и номера прерываний) гипервизору

посредством гипервызова[1]. Гипервызов – это обращение программы из ВМ к гипервизору для выполнения какой-либо операции. Синхронный характер этого обращения позволяет передавать параметры гипервызова аналогично тому, как пользовательский процесс передает параметры ядру ОС при выполнении системного вызова: числовые параметры и адреса областей памяти передаются через регистры, при необходимости гипервизор читает область памяти виртуальной машины по указанным адресам и извлекает из нее (или записывает в нее) дополнительную информацию.

Наиболее сложной является ситуация, в которой гипервизору требуется известить компоненты в виртуальной машине о некотором событии. Для этого гипервизор использует возможность, предоставляемую аппаратурой виртуализации, вбрасывать прерывания и исключительные ситуации в виртуальную машину посредством соответствующих полей в управляющей структуре ВМ[1]. После возобновления работы ВМ аппаратура обеспечивает ей доставку прерывания непосредственно перед выполнением первой инструкции в ВМ. В результате вброса прерывания ОС передает управление на обработчик данного прерывания, зарегистрированный модулем ядра в таблице обработчиков прерываний в процессе инициализации системы.

Параметры системного вызова передаются через кольцевой буфер. Буфер представляет собой замкнутый в кольцо массив структур данных (фиксированного размера), голова которого сдвигается по мере выемки запросов из буфера, а хвост — по мере помещения запросов в буфер. Если буфер переполнен, то доставка запроса откладывается до тех пор, пока в буфере не освободится место.

Структура данных, представляющая собой элемент кольцевого буфера, едина для всех системных вызовов и включает поля для всех возможных параметров фиксированной длины. Параметры переменной длины передаются через отдельный буфер переменного размера, расположенный в памяти гипервизора — хранилище. Координаты параметра переменной длины — смещение от начала хранилища и длина — специфицируются в структуре данных кольцевого буфера. Для каждого доверенного процесса гипервизор поддерживает отдельный экземпляр хранилища.

При получении запроса, содержащего параметры переменной длины, код в ВМ, которому предназначается этот запрос, выполняет гипервызов на доступ к хранилищу, передавая координаты запрашиваемого параметра и адрес буфера в собственной памяти, в который должны быть записаны данные из хранилища. Гипервизор обслуживает запрос и возобновляет выполнение ВМ. При этом он

контролирует, что границы запрашиваемого блока данных не выходят за пределы хранилища. Доступ к хранилищу возможен как по чтению, так и по записи.

Удалённое обслуживание системных вызовов сетевой подсистемы

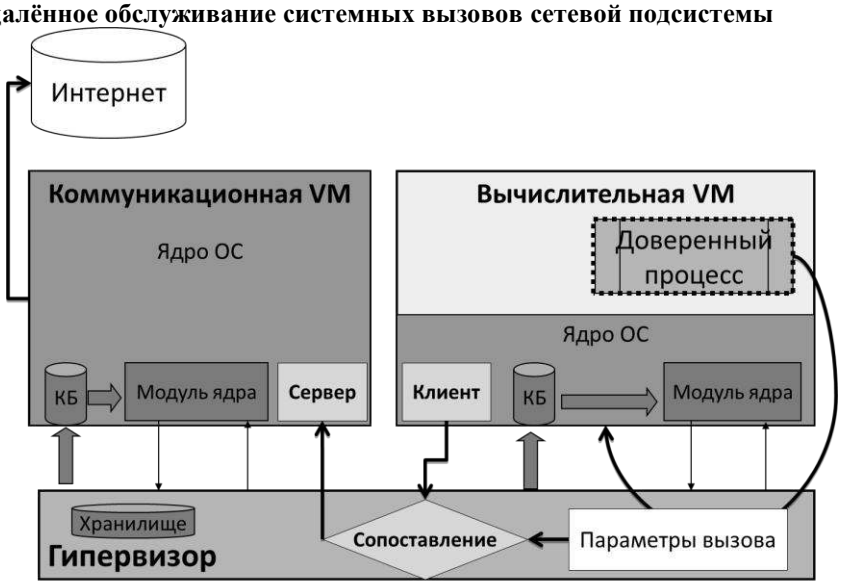

Рисунок 1. Представление компонентов системы защиты, ответственных за удалённое исполнение системных вызовов, и упрощённой схемы их взаимодействия.

Пользовательские приложения для доступа в сеть используют механизм сокетов. При создании нового сокета одним из параметров обозначается семейство протоколов. Этот параметр определяет, какие обработчики вызовет ядро ОС для корректного завершения системного вызова и создания нового сокета. В дальнейшем при записи, чтении данных и осуществлении иных действий с созданным сокетом будут вызываться обработчики в ядре ОС, связанные именно с этим семейством протоколов.

Для удалённой обработки системных вызовов, с использованием штатных механизмов ядра Linux реализуется специальный протокол, представляющий собой ещё один модуль ядра ОС. Код протокола можно разделить на клиентскую и серверную части. Серверная часть функционирует в коммуникационной ВМ, клиентская, соответственно, в вычислительной (рис.1).

При поступлении системного вызова от доверенного приложения, до того как он будет обработан ядром вычислительной ВМ, работа этой ВМ прерывается и управление передаётся гипервизору. Гипервизор подменяет идентификатор семейства протоколов в аргументе системного вызова на специальный, соответствующий новому протоколу, который является частью системы защиты. Гипервизор также копирует параметры системного вызова от доверенного приложения. Далее возобновляется работа вычислительной ВМ. Системный вызов поступает в ядро ОС. Оно анализирует параметры системного вызова и передаёт управление зарегистрированным обработчикам в клиентской части нового протокола.

В клиентской части осуществляется выделение необходимых ресурсов ядра, создание пакета, содержащего в себе аргументы системного вызова, запись его в кольцевой буфер и извещение гипервизора посредством гипервызова.

После того, как управление вновь возвращается в гипервизор происходит частичное сопоставление предварительно сохранённых аргументов системного вызова и поступивших из кольцевого буфера вычислительной ВМ. Т. к. ядро вычислительной ВМ является недоверенным, то произвести подобное сопоставление необходимо для предотвращения утечки данных по подложным адресам. В случае ошибки сопоставления поступивший системный вызов не будет исполнен в коммуникационной ВМ.

Гипервизор записывает поступившие данные в кольцевой буфер коммуникационной виртуальной машины и вбрасывает в неё прерывание. Обработчик прерывания передаёт управление серверной части протокола. Данные из кольцевого буфера распаковываются. Далее происходит стандартная обработка системного вызова с поступившими аргументами, как если бы он поступил от пользовательского приложения в ОС, работающей без системы защиты.

Сложность в возвращении результатов удалённого исполнения системного вызова связана с тем, что некоторые системные вызовы легально могут модифицировать память процесса. Гипервизор гарантирует, что модифицироваться будут только явно указанные в параметрах системного вызова диапазоны адресов. Это достигается за счёт того, что при выполнении системного вызова драйвер в ядре ОС выделяет область памяти необходимого размера и модифицирует параметры системного вызова - ОС записывает результаты в эту область памяти. При возврате из системного вызова гипервизор копирует данные в соответствующую область памяти доверенного процесса.

Заключение

В статье рассматривается подход к реализации системы защиты приложений, позволяющей избирательно предоставлять доступ к сетевым ресурсам отдельным доверенным пользовательским приложениям.

Подход основан на выполнении операционной системы внутри виртуальной машины и размещения доверенной части системы защиты в теле гипервизора. Этим достигается полный контроль доступа процессов, выполняющихся в ВМ, к аппаратным ресурсам. Система защиты при этом остаётся недоступной для атак со стороны вредоносного программного обеспечения.

Использование технологии аппаратной виртуализации для перехвата системных вызовов, чтения их параметров и записи результатов позволяет делегировать обслуживание перехваченных системных вызовов другой системе. Возможность предоставлять отдельным процессам контролируемый доступ к ресурсам, к которым сама операционная система доступа не имеет, позволяет эффективно решить отдельные аспекты задачи сохранения конфиденциальности пользовательских данных.

Библиографический список

1. AMD64 Architecture Programmer's Manual. Volume 2: System Programming. // Revision 3.17. Advanced Micro Devices. 2010.
2. R. Riley, X. Jiang, and D. Xu, "Multi-aspect profiling of kernel rootkit behavior," in EuroSys '09: Proceedings of the 4th ACM European conference on Computer systems. New York, NY, USA: ACM, 2009, pp. 47–60.
3. Joshua LeVasseur, Volkmar Uhlig, Jan Stoess, Stefan G¨otz "Unmodified Device Driver Reuse and Improved System Dependability via Virtual Machines" Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6. San Francisco, CA, 2004. Pages: 2 - 2.
4. X. Chen, T. Garfinkel, E. C. Lewis, P. Subrahmanyam, C. A. Waldspurger, D. Boneh, J. Dwoskin, and D. R. Ports, "Overshadow: a virtualization-based approach to retrofitting protection in commodity operating systems," in ASPLOS XIII: Proceedings of the 13th international conference on Architectural support for programming languages and operating systems. New York, NY, USA: ACM, 2008,pp. 2–13.
5. R. Ta-Min, L. Litty, and D. Lie, "Splitting interfaces: making trust between applications and operating systems configurable," in OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation. Berkeley, CA, USA: USENIX Association, 2006, pp. 279–292.
6. Adams, K., Agesen, O. "A comparison of software and hardware techniques for x86 virtualization." In Proceedings of the 12th international Conference on Architectural Support For Programming Languages and Operating Systems, ACM, 2006, pp. 2-13.
7. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles. New York, NY, USA: ACM, 2003, pp. 164–177.
8. Takahiro Shinagawa, Hideki Eiraku, Kouichi Tanimoto, Kazumasa Omote, Shoichi Hasegawa, Takashi Horie, Manabu Hirano, Kenichi Kourai, Yoshihiro Oyama, Eiji Kawai, Kenji Kono, Shigeru Chiba, Yasushi Shinjo, Kazuhiko Kato "BitVisor: a thin hypervisor for enforcing i/o device security". Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments. Washington, DC, USA. Pages: 121-130.