Использование аппаратной виртуализации в контексте информационной безопасности

Д. В. Силаков
silakov@ispras.ru

Аннотация. Статья посвящена возможностям использования аппаратной виртуализации для решения различных задач информационной безопасности. Предлагается обзор подходов к повышению безопасности программных систем, основанных на использовании виртуализации. Также приводится обзор возможных сценариев использования виртуализации злоумышленниками. Указываются области применения и ограничения существующих решений и дальнейшие перспективы развития рассматриваемой области.

Ключевые слова: информационная безопасность; виртуализация; гипервизор.

1. Введение

Задачами информационной безопасности (обеспечением конфиденциальности, целостности и доступности информации) человечество уделяет внимание уже не одно столетие. На сегодняшний день основными средствами обработки и передачи информации являются компьютеры и связанные с ними технологии, в особенности - глобальные сети передачи данных. Соответственно, основные исследования сосредоточены именно в области обеспечения безопасности обработки и передачи данных в электронном виде.

В наиболее распространенных на сегодняшний день системах, основанных на процессорах архитектуры Intel x86 и ее 64-битного расширения, одна из ключевых ролей в обеспечении информационной безопасности принадлежит операционной системе (ОС). Традиционно, ОС работает на наивысшем аппаратном уровне привилегий (в кольце 0) и потенциально имеет доступ ко всем другим компонентам системы. В частности, ей доступны данные и исполняемый код работающих на машине приложений, а также подконтрольны каналы связи приложений с внешним миром. Как следствие, компрометация ОС и внедрение в нее вредоносного кода представляет собой одну из основных угроз безопасности всей системы. Несмотря на то, что за время развития индустрии ИТ было разработано немало подходов к повышению качества ПО (обзор таких методов можно найти в [1]), полностью

избавиться от ошибок в таких крупных продуктах, как современные ОС общего назначения, на сегодняшний день не представляется возможным.

Обнаружение вредоносного кода, внедренного в ОС, затруднено тем, что он работает с теми же привилегиями, что и средства обнаружения и предотвращения угроз (инструменты обнаружения вторжений, антивирусы и прочие). Таким образом, вредоносные программы имеют возможность скрывать следы своей деятельности – например, манипулируя системными журналами или проводя атаки непосредственно на средства защиты [2].

Такая ситуация, когда вредоносное ПО и средства защиты системы играют в «кошки-мышки» на уровне ОС, сохранялась на протяжении долго времени и, по большому счету, сохраняется сейчас. Однако представленные в 2005-2006 годах средства аппаратной поддержки виртуализации архитектуры x86 открывают новые горизонты в этой области.

2. Аппаратная поддержка виртуализации

Изначально, аппаратная поддержка виртуализации была добавлена в процессоры Intel и AMD с целью обойти ограничения архитектуры x86, сильно усложнявшие задачу запуска нескольких виртуальных машин (ВМ) на одной физической. Согласно сформулированному еще в 1974 году критерию Попека-Голдберга [3], эффективный монитор виртуальных машин (гипервизор) может быть построен, если все инструкции, способные изменить состояние ресурсов виртуальной машины, а также инструкции, поведение которых зависит от конфигурации этих ресурсов, являются привилегированными. Соблюдение этого условия позволяет разместить ВМ на непривилегированном уровне, оставив на привилегированном только гипервизор. Выполнение привилегированных инструкций в ВМ вызывает аппаратное исключение, перехватываемое гипервизором, который эмулирует необходимое поведение.

Практика показала, что в реальных системах доля таких перехватов достаточно мала и большинство инструкций может напрямую исполнятся на аппаратуре реальной машины, что обеспечивает эффективность гипервизора в плане производительности.

Основной проблемой архитектуры x86 является наличие непривилегированных инструкций, способных изменить состояние ресурсов виртуальной машины, а также инструкций, которые ведут себя по-разному, будучи выполнены на разных уровнях привилегий. Для подобных инструкций метод перехвата аппаратных исключений не работает (поскольку никаких исключений при их выполнении на непривилегированном уровне не возникает) и необходимы другие подходы – такие, как бинарная трансляция (используемая в VMware Workstation и VirtualBox) или паравиртуализация (Xen, KVM) [4].

Технологии виртуализации от Intel (VT-x) и AMD (AMD-V) подразумевают дополнительное разграничение привилегий на аппаратном уровне. Более точно, вводятся два новых уровня привилегий – корневой (“root”) и некорневой (“nonroot”). “Старые” кольца x86 располагаются внутри некорневого уровня, а в корневом уровне может размещаться гипервизор. ВМ располагаются в некорневом уровне (при этом ОС внутри них работают в кольце 0, как на “обычной” машине). При возникновении определенных ситуаций, определяемых посредством соответствующих управляющих структур, ВМ останавливается, а управление передается гипервизору, который выполняет необходимые преобразования инструкций и их аргументов и возвращает управление ВМ.

Первые реализации аппаратной поддержки виртуализации были сосредоточены на перехвате инструкций, то есть нацеливались, прежде всего, на виртуализацию процессора. Однако со временем производители добавили средства работы с памятью и с устройствами, предоставив возможность создавать гипервизоры, способные контролировать практически все активности системы.

Важным фактом является то, что гипервизор – это обычная программа, и его функциональность не обязательно должна ограничиваться управлением виртуальными машинами. Фактически, аппаратная виртуализация позволяет запускать на физической машине программу, привилегии которой выше, чем привилегии ОС. Такая программа может останавливать работу системы в случае возникновения определенных событий, совершать произвольные манипуляции с адресным пространством ОС и ее компонентов, и возвращать ей управление. Именно на этом аспекте и основываются различные методы, как повышения безопасности программных систем, так и их компрометации, о которых пойдет речь ниже.

3. Обнаружение вредоносного ПО

Одним из способов использования нового уровня привилегий, лежащего ниже нулевого кольца x86, является помещение на этот уровень инструментов обнаружения вредоносного ПО. Такой подход существенно затрудняет возможность компрометации средств защиты со стороны кода, работающего внутри ОС. Кроме того, злоумышленникам становится сложнее скрывать следы своей деятельности, поскольку все действия с системными журналами, файлами и другими ресурсами могут перехватываться и анализироваться гипервизором.

При этом сложную функциональность по анализу событий, происходящих в системе, не обязательно реализовывать непосредственно в гипервизоре. Вместо этого можно запустить еще одну виртуальную машину, невидимую для пользователя основной системы, в которой и будет проводиться анализ. Внутри такой вспомогательной машины может быть развернута одна из массовых ОС (например, Linux), что позволит использовать уже имеющиеся

инструменты анализа системных событий и выявления вредоносных действий [5].

Помимо использования традиционных средств анализа, ряд исследователей (см., например, [6], [7]) предлагают новые способы анализа и контроля происходящих в системе событий, основанные на использовании гипервизора, а точнее – его возможности приостанавливать работу ВМ при наступлении определенных событий с последующим анализом адресного пространства и ресурсов системы.

Следует отметить, что гипервизор (и, соответственно, вспомогательная ВМ) «видит» операционную систему, ее компоненты и приложения на достаточно низком уровне и не обладает никакой семантической информацией о процессах ОС, структуре ее файловой системы и других деталях. В то же время наличие некоторых семантических сведений активно используется рядом средств защиты системы – например, инструментами обнаружения вторжений. Проблема предоставления такой информации также активно исследуется. Часть предлагаемых решений ([7], [8]) сводятся к добавлению в гипервизор возможности реконструкции данных о системе на основе анализа ее ресурсов – адресного пространства, жесткого диска и других. Альтернативой является внедрение в каждую ВМ специального агента, который бы сообщал гипервизору необходимые сведения – подобный подход используется, например, в технологии VMware VMsafe [9]. Однако в такой схеме на гипервизор ложится дополнительная задача по защите адресного пространства своих агентов внутри ВМ с целью защиты их от компрометации.

Что касается производительности, то многие исследовательские прототипы работают медленнее своих аналогов, не использующих виртуализацию, на 5-20%, что в большинстве случаев является вполне приемлемым. Важно отметить, что инструменты, работающие на уровне гипервизора, могут работать быстрее своих аналогов внутри ВМ – например, подтверждающие этот факт экспериментальные данные по ряду известных антивирусов приведены в [8]. Этот аспект важен, в частности, в контексте активно развивающихся облачных вычислений, где уже существуют соответствующие промышленные решения – так, инструменты Catbird могут быть использованы клиентами Amazon EC2 для аудита своих систем. Пионер и один из лидеров в области виртуализации архитектуры x86 – компания VMware – разработала уже упоминавшуюся технологию VMsafe, упрощающую разработку и использование средств обеспечения безопасности в виртуализированных средах, построенных на продуктах VMware, посредством предоставления API для доступа к гипервизору.

4. Внедрение вредоносного ПО

Возможность иметь программу, неподконтрольную ОС, заинтересовала исследователей в области безопасности не только с точки зрения создания программ, обнаруживающих вредоносные действия, но и с точки зрения

осуществления этих самых действий - ведь теоретически, вредоносное ПО, расположенное на корневом уровне привилегий, крайне сложно обнаружить с использованием средств защиты изнутри ОС.

Основная трудность использования такого способа компрометации заключается во внедрении гипервизора в систему. Как Intel, так и AMD предлагают возможность запуска гипервизора из уже работающей системы посредством специальных инструкций SENTER [10] и SKINIT [11] соответственно. Эти инструкции осуществляют приостановку работающей на машине системы и запускают некоторую программу — например, гипервизор. После чего система продолжает работу, но уже внутри ВМ, управляемой гипервизором. Для ОС и приложений внутри нее запуск гипервизора проходит прозрачно (при условии, что гипервизор не закроет доступ к каким-либо ресурсам и устройствам, используемым системой, не предоставив виртуализированного аналога).

Целью Intel TXT и AMD SVM является предоставление возможности запускать доверенное ПО в скомпрометированной системе (именно запуска — защита ПО в процессе работы не осуществляется). Реализуемая схема запуска программ получила название «поздний запуск» («late launch»). Естественно, выполнение такого запуска может осуществляться только из привилегированного кольца 0 – то есть, как правило, от лица ОС. Поэтому для прозрачного (скрытого от пользователя) внедрения гипервизора необходимо использовать какую-либо брешь в ОС, либо рассчитывать на некомпетентные действия администратора системы, которые приведут к запуску гипервизора. К тому же, запустить гипервизор один раз мало – необходимо предусмотреть его запуск при перезагрузке системы. Таким образом, применение виртуализации для компрометации системы не избавляет от необходимости обращаться к «традиционным» способам атаки.

Впрочем, как и практически все программное обеспечение, реализации Intel TXT и AMD SVT могут содержать уязвимости, дающие возможность их компрометации. Например, одна из уязвимостей, подтвержденная (и, естественно, исправленная) специалистами Intel описана в работе [12].

Также стоит упомянуть о подходах к компрометации системы с помощью атаки на System Management Mode (SMM) – специальный режим работы процессора, в котором приостанавливается работа всей системы (включая гипервизор, если он есть), и запускается программа-обработчик из аппаратной прошивки (какой именно обработчик – определяется причиной, по которой был осуществлен переход в SMM). Соответственно, компрометация программы, работающей в SMM, потенциально ведет к компрометации всей системы. Несмотря на то, что SMM появился еще в процессорах Intel 386, подходы к его компрометации появились лишь в последние годы [13], [14] – практически совпав с началом исследований возможности применения виртуализации для компрометации системы.

Наконец, компрометации могут быть подвергнуты и другие низкоуровневые компоненты машины — например, BIOS [15] или Intel Active Management Technology [16].

В настоящее время использование виртуализации (равно как и SMM и других низкоуровневых компонентов) для компрометации системы существует скорее в виде исследовательских работ. Существуют инструменты, демонстрирующие возможность соответствующих атак на систему – из наиболее известных можно выделить BluePill [17], а также SubVirt, разработанный в Microsoft Research [18]. Однако до массового применения подобных средств дело еще не дошло. Не в последнюю очередь это обусловлено техническими сложностями, а также сильной зависимостью от аппаратного обеспечения, используемого на целевой системе [19].

Кроме того, несмотря на сложность обнаружения вредоносного гипервизора из ОС, эта задача не является абсолютно неразрешимой – точнее, существуют подходы к определению того, работает ли ОС на «голом» железе или внутри виртуальной машины. Соответственно, если выполнение ОС в виртуальной машине не является легитимным, то администратор системы должен принять меры по выявлению и устранению гипервизора. Большинство подходов основано на том факте, что работа гипервизора, так или иначе, замедляет работу системы. Даже если гипервизор будет манипулировать часами, которые видит ОС, всегда есть возможность воспользоваться внешними (по отношению к системе) часами [20].

Наконец, если пользователю не нужна поддержка аппаратной виртуализации, ее можно просто отключить средствами BIOS. А если виртуализация используется, то в момент атаки на машине уже может работать гипервизор, вытеснение которого вредоносным монитором будет, очевидно, замечено пользователем. Обработка ситуаций, когда «виртуализировать» надо не только работающую ОС, но еще и гипервизор, на данный момент изучена достаточно слабо, и до реальных применений имеющихся теоретических разработок дело пока не дошло.

Впрочем, если в целевой системе уже работает какой-либо монитор виртуальных машин, то создавать и внедрять собственный гипервизор не обязательно. Можно провести атаку на уже существующий, получив в случае успеха доступ ко всем виртуальным машинам. Компрометация существующего гипервизора даже более выгодна для злоумышленника, чем внедрение собственного – ведь наличие существующего гипервизора легитимно (нет необходимости пытаться скрыть его существование от пользователя) и его запуск, вероятно, будет осуществляться и после перезапуска машины.

Возможность компрометации гипервизора допускалась еще разработчиками первого коммерческого монитора – CP-67, работавшего на платформах IBM System/360-370. Однако отмечалось, что поскольку код гипервизора существенно меньше и проще, чем код операционной системы, то и

уязвимостей в нем существенно меньше [21]. Тем не менее, в последователе CP-67 – гипервизоре VM/370 [22] – было найдено несколько критических уязвимостей.

В IBM этот опыт учли, и при разработке последующих гипервизоров безопасности уделялось особое внимание. В частности, уже для KVM/370 использовались формальные методы для спецификации и верификации гипервизора – возможность и целесообразность применения таких методов обусловлена, опять же, относительно небольшим размером кода. Современный PR/SM – монитор для IBM System z, считающийся одним из самых безопасных гипервизоров – сертифицирован по уровню 5 Common Criteria Evaluation Assurance (EAL5) и уровню E4 ITSEC.

К сожалению, гипервизорам для массовой архитектуры x86 (таким как Xen, KVM или VMware ESX) до таких высот еще далеко, несмотря на наличие различных подходов к повышению качества подобного ПО (в том числе с использованием формальных методов – см., например, [23]). Причем, по мнению исследователей из IBM, проблемы в этих гипервизорах лежат не просто в недостаточно качественной реализации, а на уровне архитектуры [24]. Неудивительно, что время от времени в существующих решениях виртуализации обнаруживаются уязвимости, позволяющие скомпрометировать гипервизор и получить контроль над виртуальными машинами – из наиболее известных исследований, стоит выделить работы [25], [26] и [27]. Следует иметь в ввиду возможность компрометации такого рода и не считать гипервизор заведомо неуязвимым.

5. Изоляция приложений

Еще одним направлением, для которого аппаратная виртуализация открывает новые перспективы, является изолированное выполнение приложений и защита адресного пространства приложения от несанкционированного доступа. В принципе, этой задачей должна заниматься операционная система. Однако современные ОС достаточно крупны, сложны, и в них постоянно обнаруживаются уязвимости, используемые злоумышленниками в недобросовестных целях.

Изолировать приложения друг от друга пользователь ПК может посредством запуска отдельной виртуальной машины для каждого приложения (или группы приложений) – современные средства виртуализации позволяют делать это достаточно просто. Однако использование аппаратной виртуализации позволяет перенести задачу изоляции адресного пространства приложения в гипервизор, который может гарантировать изолированность не только от других приложений, но и от компонентов ОС – подобный подход, например, используется в технологии Overshadow [28], предложенной исследователями из VMware совместно с учеными университетов Стэнфорда и Принстона и Массачусетского технологического института.

Размер кода гипервизора, необходимого для выполнения задач изоляции, существенно меньше размера кода современных ОС общего назначения. В частности, в таком гипервизоре отсутствует необходимость в драйверах различных устройств, а именно драйвера являются одной из наиболее уязвимых частей современных монолитных ОС. Как было показано в предыдущем разделе, малый размер кода сам по себе не гарантирует отсутствия ошибок. Тем не менее, многие исследователи активно эксплуатируют гипотезу о том, количество уязвимостей в относительно небольшом гипервизоре существенно меньше, чем в сложной ОС общего назначения.

Как и в случае с реализацией средств обнаружения вредоносного ПО, при необходимости предоставления гипервизором какой-либо дополнительной функциональности (например, передачи управления доступом в сеть от ОС гипервизору), такая функциональность может быть реализована не в самом гипервизоре, а во вспомогательной виртуальной машине. При этом вспомогательная ВМ может полностью располагаться в оперативной памяти и быть невидимой для пользователя основной ВМ. Такой подход используется в работах, проводимых в ИСП РАН для создания среды запуска приложений в недоверенной ОС [29], [30].

Ряд исследований совмещают запуск отдельных ВМ для различных групп приложений и контроль над действиями приложений и компонентов ОС со стороны гипервизора. Одной из наиболее интересных разработок в этой области является операционная система Qubes [31]. В основе системы лежит разбиение приложений на изолированные домены, каждый из которых работает в отдельной ВМ (управляемой Linux). В основе Qubes лежит гипервизор Xen. В целях повышения безопасности коды непосредственно гипервизора и привилегированного домена (Dom0) сохраняются максимально простыми, а практически все драйвера (в том числе средства работы с сетью и с дисковыми накопителями) вынесены в отдельные вспомогательные домены.

В ОС Qubes используется специализированный X-сервер, позволяющий отображать окна приложений из всех доменов на экране пользователя и осуществлять обмен данными (“copy-paste”) между приложениями различных доменов. В итоге у пользователя складывается ощущение, что он работает с одним экземпляром ОС Linux.

По состоянию на сегодняшний день, ОС Qubos находится в состоянии разработки и не готова для промышленного использования — равно как и большинство прототипов, разработанных в этой области.

6. Заключение

Реализация аппаратной поддержки виртуализации в массовых процессорах дала толчок большому числу исследований по возможностям применения виртуализации в контексте информационной безопасности — как в области

защиты информации, так и в области компрометации программных систем. Однако, несмотря на достаточно большое количество теоретических работ и прототипов, говорить о массовом использовании виртуализации для решения проблем безопасности пока рано.

Одной из основных причин такого положения является техническая сложность реализации предлагаемых решений и зависимость от аппаратной платформы — несмотря на схожесть концепций, реализации Intel и AMD имеют ряд серьезных отличий. К тому же, история аппаратной поддержки виртуализации насчитывает немногим более пяти лет, на протяжении которых технология активно дорабатывалась и расширялась. Многие работы, предлагающие использовать виртуализацию для решения различных задач, опираются на второе поколение аппаратной виртуализации (Intel Extended Page Table, AMD Rapid Virtualization Indexing), представленное в процессорах только в 2009-2010 годах. Возможно, через несколько лет — по мере увеличения доли процессоров с поддержкой виртуализации на машинах пользователей — мы увидим и рост числа приложений, использующих виртуализацию для решения задач информационной безопасности.

Литературы

[1] Липаев В.В. Методы обеспечения качества крупномасштабных программных средств. М.: СИНТЕГ, 2003. 520 с.

[2] T. Garfinkel. Traps and Pitfalls: Practical Problems in System Call Interposition Based Security Tools. // Proc. of the Internet Society's 2003 Symposium on Network and Distributed System Security. 2003. Pp. 163-176.

[3] G.J. Popek, R.P. Goldberg. Formal Requirements for Virtualizable Third Generation Architectures. // Communications of the ACM, Volume 17, Issue 7, July 1974, pp. 412-421.

[4] Касперски К. Аппаратная виртуализация или эмуляция "без тормозов". // InsidePro, 2007. [HTML] http://www.insidepro.com/kk/159/159r.shtml

[5] A. Dinaburg, P. Royal, M. Sharif, W. Lee. Ether: Malware Analysis via Hardware Virtualization Extensions. // Proc. of the 15th ACM conference on Computer and communications security. 2008. Pp. 51-62.

[6] S. Krishnan, K.Z. Snow, F. Monrose. Trail of bytes: efficient support for forensic analysis. // Proc. of the 17th ACM conference on Computer and communications security. 2010. Pp. 50-60.

[7] T. Garfinkel, M. Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. // Proc. of the Symposium on Network and Distributed System Security (NDSS'03). 2003. [PDF] http://suif.stanford.edu/papers/vmi-ndss03.pdf

[8] X. Jiang, X. Wang, D. Xu. Stealthy Malware Detection Through VMM-Based “Out-of-the-Box” Semantic View Reconstruction. // Proc. of the 14th ACM conference on Computer and communications security. 2007. Pp. 128-138.

[9] New VMware VMsafe Technology Allows the Virtual Datacenter to Be More Secure Than Physical Environments. // Press release. 2007. [HTML] http://www.vmware.com/company/news/releases/vmsafe_vmworld.html

[10] Intel Trusted Execution Technology Architectural Overview. // Intel White Paper. 2008. [PDF] http://www.intel.com/technology/security/downloads/arch-overview.pdf

[11] G. Strongin. Trusted Computing Using AMD «Pacifica» and «Precidio» Secure Virtual Machine Technology. // Information Security Technical Report. 2005. Volume 10, Issue 2, pp. 120-132.

[12] R. Wojtczuk, J. Rutkowska, A.Tereshkin. Another Way to Circumvent Intel Trusted Execution Technology. // Invisible Things Lab. December, 2009. [PDF] http://invisiblethingslab.com/resources/misc09/Another%20TXT%20Attack.pdf

[13] R. Wojtczuk, J. Rutkowska. Attacking Intel Trusted Execution Technology. // Black Hat DC 2009. [PDF] http://invisiblethingslab.com/resources/bh09dc/Attacking%20Intel%20TXT%20-%20paper.pdf

[14] S. Embleton, S. Sparks, C. Zou. SMM Rootkits: A New Breed of OS Independent Malware. // Proc. of the 4th international conference on Security and privacy in communication networks. Istanbul, Turkey, 2008. Article #11, pp. 1-12.

[15] R. Wojtczuk, A. Tereshkin. Attacking Intel BIOS. // Black Hat USA 2009. [PDF] http://invisiblethingslab.com/resources/bh09usa/Attacking%20Intel%20BIOS.pdf

[16] R. Wojtczuk, A. Tereshkin. Introducing Ring -3 Rootkits. // Black Hat USA 2009. [PDF] http://invisiblethingslab.com/resources/bh09usa/Ring%20-3%20Rootkits.pdf

[17] J. Rutkowska. Subverting Vista Kernel For Fun And Profit. // Black Hat USA 2006. [PDF] http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Rutkowska.pdf

[18] S.T. King, P.M. Chen, Y.M. Wang, C. Verbowski, H.J. Wang, J.R. Lorch. SubVirt: Implementing malware with virtual machines. // Proc. of the 2006 IEEE Symposium on Security and Privacy. 2006. Pp. 314–327.

[19] A. Liguori. Debunking Blue Pill myth. // Interview to Virtualization.info. August, 2006. [HTML] http://virtualization.info/en/news/2006/08/debunking-blue-pill-myth.html

[20] T. Garfinkel, K. Adams, A. Warfield, J. Franklin. Compatibility is Not Transparency: VMM Detection Myths and Realities. // 11th Workshop on Hot Topics in Operating Systems (HotOS-XI), 2007. [PDF] http://www.stanford.edu/~talg/papers/HOTOS07/vmm-detection-hotos07.pdf

[21] S.E. Madnick, J.J. Donovan. Application and analysis of the virtual machine approach to information system security and isolation. // Proc. of the workshop on virtual computer systems. ACM, 1973, pp. 210-224.

[22] C.R. Attanasio, P. W. Markstein, Ray J. Phillips. Penetrating an Operating System: A Study of VM/370 Integrity. // IBM Systems Journal, Volume 15, 1976. Pp. 102-116.

[23] И.Б. Бурдонов, А.С. Косачев, В.В. Кулямин. Безопасность, верификация и теория конформности. // Материалы Второй международной научной конференции по проблемам безопасности и противодействия терроризму. Москва, МНЦМО, 2007. С. 135-159.

[24] P.A. Karger, T.J. Watson. Is Your Virtual Machine Monitor Secure? // Materials of Third Asia-Pacific Trusted Infrastructure Technologies Conference, 2008. Pp. 5-5.

[25] T. Garfinkel, M. Rosenblum. When Virtual is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments. // 10th Workshop on Hot Topics in Operating Systems (HotOS-X), 2005. [PDF] http://www.stanford.edu/~talg/papers/HOTOS05/virtual-harder-hotos05.pdf

[26] J. Rutkowska. Security Challenges in Virtualized Environments. // RSA Conference, 2008. [PDF] http://www.invisiblethingslab.com/resources/rsa08/Security%20Challanges%20in%20Virtualized%20Enviroments%20-%20RSA2008.pdf

[27] R. Wojtczuk. Subverting the Xen Pypervisor. // Black Hat USA 2008. [PDF] http://invisiblethingslab.com/resources/misc08/xenfb-adventures-10.pdf

[28] X. Chen, T. Garfinkel, E.C. Lewis, P. Subrahmanyam, C.A. Waldspurger, D. Boneh, J. Dwoskin, D.R.K. Ports. Overshadow: A Virtualization-Based Approach to Retrofitting Protection in Commodity Operating Systems. // Proc. of the 13th international conference on Architectural support for programming languages and operating systems (ASPLOS XIII). 2008. Pp. 2-13.

[29] Яковенко П.Н. Прозрачный механизм удаленного обслуживания системных вызовов. // Труды Института системного программирования РАН. Том 18. 2010. С. 221-241.

[30] I. Burdonov, A. Kosachev, P. Iakovenko. Virtualization-Based Separation of Privilege: Working With Sensitive Data In Untrusted Environment. // Proc. Of the 1st EuroSys Workshop on Virtualization Technology for Dependable Systems. 2009. Pp. 1-6.

[31] Qubes Architecture Specification. Version 0.3. 2010. [PDF] http://qubes-os.org/files/doc/arch-spec-0.3.pdf