УДК 004.45

Машков К.А.,
студент,
4 курс, направление «Прикладная информатика»
Северо-Кавказский федеральный университет
Россия, г. Ставрополь

ИССЛЕДОВАНИЕ МОДЕЛЕЙ ИЗОЛЯЦИЦИ КОНТЕЙНЕРОВ И ВИРТУАЛЬНЫХ МАШИН НА УРОВНЕ ЯДРА ОПЕРАЦИОННОЙ СИСТЕМЫ

Аннотация: *В статье рассматриваются модели изоляции контейнеров и виртуальных машин на уровне ядра операционной системы. Целью работы является сравнительный анализ механизмов, формирующих границы изоляции, а также оценка их сильных и слабых сторон с позиции безопасности, производительности и практической применимости в современных вычислительных средах. Сделан вывод о том, что выбор технологии должен определяться моделью угроз, требованиями к плотности размещения и допустимой ценой компромисса между безопасностью и производительностью.*

Ключевые слова: *контейнеры, виртуальные машины, изоляция, ядро операционной системы, namespaces, KVM, gVisor, Kata Containers.*

Abstract: *The article examines container and virtual machine isolation models at the operating system kernel level. The aim of the study is to provide a comparative analysis of the mechanisms that establish isolation boundaries, as well as to assess their strengths and weaknesses in terms of security, performance, and practical applicability in modern computing environments. It is concluded that the choice of technology should be determined by the threat model, density requirements, and the acceptable trade-off between security and performance.*

Keywords: *containers, virtual machines, isolation, operating system kernel, namespaces, KVM, gVisor, Kata Containers.*

Развитие облачных платформ, edge-инфраструктуры и DevOps-подходов привело к широкому использованию двух базовых способов изоляции вычислительных нагрузок: контейнеризации и аппаратно поддерживаемой виртуализации. Обе технологии решают сходную прикладную задачу - запуск независимых приложений в одном физическом окружении, - однако реализуют изоляцию на разных уровнях системного стека.

Для контейнеров характерна модель изоляции на уровне ядра операционной системы: процессы разделяют одно ядро хоста, а необходимые границы формируются с помощью пространств имен, контрольных групп, политик доступа и фильтрации системных вызовов. Виртуальные машины, напротив, используют гипервизор и виртуализированное оборудование, благодаря чему каждая гостевая система получает собственное ядро и собственное системное окружение.

Актуальность исследования обусловлена тем, что в современных инфраструктурах выбор между контейнерами и виртуальными машинами больше не сводится к вопросу удобства развертывания. Он напрямую связан с моделью угроз, требованиями к среде с вариацией уровня доступа, рисками container escape и допустимой стоимостью защитных мер. Кроме того, в последние годы появились гибридные решения, например Kata Containers и gVisor, которые стремятся объединить преимущества обеих моделей.

Теоретические основы изоляции на уровне ядра ОС

Под изоляцией на уровне ядра операционной системы понимается совокупность механизмов, ограничивающих влияние одного процесса, группы процессов или гостевой среды на другие вычислительные объекты и на хостовую систему. В контексте контейнеризации такая изоляция строится

вокруг одного ядра Linux, тогда как в виртуализации ядро участвует уже как часть гипервизорной подсистемы, предоставляющей виртуальные машины с эмулируемыми или паравиртуализированными устройствами.

Для контейнеров ключевыми механизмами являются пространства имен, которые создают отдельные представления системных ресурсов: идентификаторов процессов, сетевых интерфейсов, точек монтирования, хоста, межпроцессного взаимодействия и пользовательских идентификаторов [1]. За счет этого контейнер получает собственную «видимость» системы, хотя фактически работает в общем ядре с другими контейнерами. Дополнительно контрольные группы ограничивают и учитывают использование процессорного времени, памяти, ввода-вывода и других ресурсов, снижая риск взаимного истощения ресурсов и повышая предсказуемость работы [2].

Однако пространства имен и контрольные группы сами по себе не устраняют главный структурный риск контейнеризации - наличие общего ядра. Если атакующий получает возможность эксплуатации уязвимости ядра или ошибки конфигурации runtime, граница между контейнером и хостом может быть нарушена. Поэтому практическая контейнерная изоляция усиливается дополнительными механизмами: seccomp ограничивает набор допустимых системных вызовов, LSM-подсистемы (SELinux, AppArmor) вводят политики доступа, а user namespaces позволяют отображать пользователя root внутри контейнера в непривилегированный идентификатор на стороне хоста [1, 2].

Виртуальные машины реализуют иной принцип. На стороне хоста гипервизор, например KVM в Linux, использует аппаратную поддержку виртуализации процессора и памяти для создания изолированных гостевых окружений. Каждая виртуальная машина запускает собственное ядро и собственное пользовательское пространство, а взаимодействие с оборудованием проходит через слой виртуализации [3]. Это означает, что компрометация гостевой ОС не тождественна компрометации хоста: для

выхода за пределы гостя требуется уже эксплуатация ошибок гипервизора, эмуляции устройств или управляющей плоскости виртуализации.

Сравнительный анализ контейнеров и ВМ как моделей изоляции

С точки зрения архитектурной границы доверия контейнеры и виртуальные машины различаются принципиально. У контейнеров доверенной вычислительной базой является ядро хоста и весь стек контейнерного runtime. У виртуальных машин доверенная база включает гипервизор, модули виртуализации и часть управляющей инфраструктуры, но не требует совместного использования одного ядра между изолируемыми нагрузками. Следовательно, у контейнеров выше чувствительность к ошибкам ядра, а у виртуальных машин - к ошибкам гипервизора и эмуляции устройства.

По производительности контейнеры обычно оказываются более легковесными. Они быстрее запускаются, требуют меньше памяти на одно изолированное окружение и позволяют плотнее размещать сервисы на одном узле. Это связано с отсутствием отдельного гостевого ядра и с минимизацией накладных расходов на виртуализацию оборудования. Виртуальные машины, напротив, требуют выделения ресурсов под гостевую ОС и инициализацию полноценного виртуального окружения, из-за чего их запуск и обслуживание обходятся дороже.

С точки зрения безопасности прямое утверждение о безусловном превосходстве виртуальных машин было бы упрощением. На практике уровень защищенности зависит от настроек платформы, политики обновлений, использования mandatory access control, корректности образов и минимизации привилегий. Тем не менее именно ВМ обычно рассматриваются как более сильная граница изоляции для недоверенных или разноуровневых нагрузок, поскольку они не разделяют с соседями одно ядро и тем самым уменьшают риск массового компрометирования через kernel exploit [4].

Контейнеры особенно эффективны в сценариях, где критичны скорость масштабирования, воспроизводимость окружения и высокая плотность

размещения. Они хорошо подходят для микросервисной архитектуры, CI/CD-конвейеров и внутренних платформ с управляемой моделью доверия. Виртуальные машины целесообразны там, где необходима жесткая изоляция арендаторов, поддержка разных гостевых ОС, выполнение унаследованных систем или соответствие строгим требованиям безопасности и комплаенса.

Таким образом, контейнеры и виртуальные машины следует рассматривать не как взаимоисключающие альтернативы, а как инструменты с разной глубиной изоляции. Для инженерного выбора важно сравнивать не только производительность, но и класс атак, от которых требуется защита: lateral movement между сервисами, container escape, компрометация плоскости оркестрами, атаки на гипервизор и побочные каналы.

Современные подходы к усилению изоляции

Развитие инфраструктурных платформ привело к появлению промежуточных решений, стремящихся уменьшить главный недостаток контейнеров - общий kernel attack surface - без полного отказа от контейнерной модели эксплуатации. Одним из таких подходов является Kata Containers, где каждый контейнер или pod запускается внутри легковесной виртуальной машины. В результате привычный контейнерный подход сохраняется, но между рабочей нагрузкой и хостом появляется дополнительный уровень аппаратно поддерживаемой изоляции.

Другой подход реализован в gVisor. В отличие от классического контейнера, приложение взаимодействует не напрямую с ядром хоста, а с пользовательским слоем-песочницей, реализующим значительную часть Linux-интерфейса. Это уменьшает набор системных вызовов, достигающих ядра хоста, и тем самым сокращает поверхность атаки. Такой подход особенно интересен для запуска недоверенного кода, однако он может сопровождаться дополнительными ограничениями совместимости и производительности.

В виртуализации важным направлением стало развитие приватных вычислений (confidential computing). Например, технология AMD Secure

Encrypted Virtualization (SEV) позволяет шифровать память виртуальной машины отдельным ключом, что снижает риски чтения содержимого гостя со стороны гипервизора или иных сущностей платформы. На практике это не заменяет традиционные меры защиты, но усиливает модель изоляции в среде с вариацией уровня доступа и в сценариях обработки чувствительных данных.

Практические рекомендации по выбору модели изоляции

При выборе модели изоляции прежде всего необходимо формализовать модель угроз. Если нагрузки принадлежат одной доверенной организации, а основными критериями выступают скорость поставки, масштабирование и плотность размещения, рациональным выбором являются контейнеры с корректно настроенными пространствами имен, группами, профилями приватных вычислений и политиками LSM. В таком случае усиление изоляции достигается не заменой технологии, а жесткой конфигурацией runtime, запретом привилегированных контейнеров, ограничением прав и контролем образов.

Если же на одной инфраструктуре исполняются разноуровневые либо внешние нагрузки, в том числе недоверенный код, предпочтительнее использовать виртуальные машины или гибридные подходы. Для публичных облаков, образовательных платформ с выполнением пользовательских программ, CI-ферм и сервисов shared hosting более сильная граница изоляции часто оказывается важнее выигрыша в плотности размещения.

Отдельно следует учитывать эксплуатационные компромиссы. Контейнерная платформа требует зрелого управления образами, политиками безопасности и обновлением ядра хоста, тогда как виртуализация повышает требования к управлению гипервизором, сетевой конфигурации и образам гостевых ОС.

Таблица 1.

Сравнительная характеристика моделей изоляции

Критерий	**Контейнеры**	**Виртуальные машины**	**Гибридные модели**
Граница изоляции	Общее ядро хоста; логическая изоляция процессов и ресурсов	Отдельное гостевое ядро и виртуализированное оборудование	Контейнерный интерфейс + дополнительный sandbox или легковесная ВМ
Поверхность атаки	Kernel attack surface хоста, runtime, образы и конфигурация	Гипервизор, эмуляция устройства, управляющая плоскость	Часть риска ядра снижается за счет промежуточного слоя
Производительность	Высокая плотность размещения и быстрый старт	Большие накладные расходы и медленный запуск	Компромисс между изоляцией и накладными расходами
Типовые сценарии	Микросервисы, CI/CD, внутренние платформы	Среды с вариацией уровней доступа, недоверенные нагрузки, устаревшие ОС	Публичные сервисы-песочницы, чувствительные контейнерные нагрузки

Заключение

Проведенное исследование показало, что контейнеры и виртуальные машины реализуют разные по глубине и принципам модели изоляции на уровне ядра операционной системы. Контейнеры используют пространства имен, контрольные группы и дополнительные политики ограничения доступа, обеспечивая легковесную и масштабируемую изоляцию при совместном использовании общего ядра. Виртуальные машины строят более жесткую границу за счет гипервизора и собственного гостевого ядра, уменьшая зависимость изоляции от безопасности общего ядра хоста.

Установлено, что преимущества контейнеров проявляются в высокой плотности размещения, скорости запуска и удобстве интеграции в

современные процессы разработки и эксплуатации. Преимущества виртуальных машин связаны прежде всего с более сильной изоляцией, поддержкой гетерогенных ОС и лучшей пригодностью для сценариев с вариацией уровня доступа с недоверенными нагрузками.

Дополнительно показано, что развитие гибридных технологий, таких как Kata Containers, gVisor и confidential VM, отражает современную тенденцию к многоуровневой изоляции. Это подтверждает, что практический выбор архитектуры должен основываться не на абстрактном противопоставлении технологий, а на сопоставлении конкретной модели угроз, требований к производительности и зрелости эксплуатационных.

Список использованных источников:

1. Linux Kernel Documentation. Namespaces. https://docs.kernel.org/admin-guide/namespaces/index.html (дата обращения: 14.04.2026)

2. Linux Kernel Documentation. Control Groups v2. https://docs.kernel.org/admin-guide/cgroup-v2.html (дата обращения: 14.04.2026)

3. Linux Kernel Documentation. KVM API Documentation. https://www.kernel.org/doc/html/latest/virt/kvm/index.html (дата обращения: 14.04.2026)

4. Ugale S., Potgantwar A. Container Security in Cloud Environments: A Comprehensive Analysis and Future Directions for DevSecOps // Eng. Proc., 2023.

5. Wiegratz J. A. Comparing Security and Efficiency of WebAssembly and Linux Containers in Kubernetes Cloud Computing // arXiv, 2024.