УДК 004.056

ПОДХОД К ОБЕСПЕЧЕНИЮ БЕЗОПАСНОСТИ И ИЗОЛЯЦИИ РАБОТЫ ПРИЛОЖЕНИЙ В ОПЕРАЦИОННОЙ СИСТЕМЕ LINUX

Гречин Т.П. [1]
Научный руководитель – Руссу В.Ю.[1]
[1]Военно-космическая академия имени А.Ф.Можайского
vka@mil.ru

Введение

Современные операционные системы семейства Linux широко применяются в корпоративных, промышленных и специализированных средах. Рост числа прикладных и пользовательских программ, эксплуатируемых в рамках одной системы, усиливает требования к изоляции приложений и контролю их доступа к системным ресурсам. Особенно актуальной эта задача становится в условиях использования защищённых дистрибутивов, где внедрение сторонних платформ виртуализации и контейнеризации может быть ограничено регламентами безопасности.

В настоящее время контейнеризация является доминирующим подходом к изоляции приложений. Решения данного класса активно применяются в серверной инфраструктуре и облачных средах [3, 4]. Однако их архитектура предполагает наличие дополнительных управляющих компонентов и фоновых сервисов, что увеличивает доверенную вычислительную базу и усложняет анализ безопасности. В специализированных средах такой уровень сложности не всегда оправдан.

Ядро Linux изначально предоставляет механизмы, позволяющие реализовать изоляцию на уровне файловой системы, процессов и пространств имён без привлечения внешних платформ [1, 5, 6]. Тем не менее в большинстве случаев данные средства рассматриваются как вспомогательные инструменты, а не как самостоятельная архитектурная основа для построения изолированной среды выполнения.

Целью настоящей работы является разработка и анализ подхода к изоляции пользовательских приложений в Linux, основанного исключительно на стандартных механизмах операционной системы и ориентированного на применение в защищённых дистрибутивах.

Основная часть

Предлагаемый подход основывается на принципе минимально достаточной изоляции и отказе от использования специализированных контейнерных платформ. Архитектура решения строится на композиции базовых механизмов Linux: chroot-окружений, управляемых точек монтирования, разграничения прав доступа и запуска процессов от имени непривилегированного пользователя.

Файловая изоляция реализуется путём формирования отдельного корневого окружения, внутри которого размещаются только необходимые зависимости приложения. Доступ к ресурсам хост-системы предоставляется выборочно через управляемые bind-монтирования. Такой подход позволяет явно контролировать перечень доступных каталогов и предотвращает несанкционированный доступ к системным файлам.

Контроль доступа к устройствам организуется через выборочный проброс узлов пространства /dev. Приложению предоставляются только те аппаратные ресурсы, которые необходимы для его функционирования. Это соответствует принципу наименьших привилегий и снижает потенциальную поверхность воздействия.

Особое внимание уделено управлению жизненным циклом приложения. Запуск осуществляется в контексте непривилегированного пользователя, что ограничивает

возможные последствия некорректного поведения программы. После завершения работы выполняется последовательная очистка среды: размонтирование файловых систем и отзыв временно предоставленных разрешений. Это обеспечивает воспроизводимость состояния системы при повторных запусках.

Интеграция с графической и мультимедийной подсистемами осуществляется через существующую инфраструктуру хост-системы без развёртывания дополнительных сервисов. Несмотря на определённые ограничения модели изоляции, такой компромисс позволяет сохранить работоспособность современных пользовательских приложений.

Практическая апробация подхода показала, что изолированная среда обеспечивает стабильный запуск и функционирование приложений в локальных сценариях эксплуатации без использования контейнерных платформ. При этом архитектура остаётся прозрачной и поддаётся анализу, что важно для применения в защищённых дистрибутивах, включая Astra Linux [7].

Выводы

В работе представлен подход к изоляции приложений в операционной системе Linux, основанный на использовании стандартных механизмов ОС без привлечения специализированных контейнерных решений.

Показано, что при корректной архитектурной композиции базовые средства Linux позволяют сформировать управляемую и воспроизводимую среду выполнения приложений. Предлагаемое решение минимизирует доверенную вычислительную базу, упрощает анализ архитектуры и может быть использовано в системах с повышенными требованиями к безопасности.

Подход ориентирован на локальные сценарии эксплуатации и не рассматривается как замена контейнеризации в распределённых инфраструктурах. Его практическая значимость заключается в возможности применения в защищённых и специализированных дистрибутивах, где использование сложных программных платформ ограничено нормативными требованиями.

Литература

1. Love R. Linux Kernel Development. 3rd ed. Boston: Addison-Wesley Professional, 2010. 480 p.
2. Tanenbaum A. S., Bos H. Modern Operating Systems. 4th ed. Upper Saddle River: Pearson Education, 2015. 1136 p.
3. Merkel D. Docker: Lightweight Linux Containers for Consistent Development and Deployment // Linux Journal. 2014. No. 239.
4. Rosen R. Linux Containers and the Future Cloud // Linux Journal. 2014. No. 240.
5. Linux Foundation. Namespaces in Linux [Электронный ресурс]. URL: https://man7.org/linux/man-pages/man7/namespaces.7.html (дата обращения: 24.02.2026).
6. Linux Foundation. mount(2) – Linux manual page [Электронный ресурс]. URL: https://man7.org/linux/man-pages/man2/mount.2.html (дата обращения: 24.02.2026).
7. Astra Linux. Документация по архитектуре и механизмам безопасности ОС Astra Linux. М.: РусБИТех-Астра, 2023.