Научная статья/Scientific article
УДК 004.056.5; 004.4'24; 004.492.2
http://dx.doi.org/10.26583/bit.2026.4.12
https://elibrary.ru/wgiezk

ПРИНЦИПЫ ПОСТРОЕНИЯ ПАТТЕРНОВ БЕЗОПАСНОСТИ ДЛЯ АГЕНТСКИХ СЕРВИСОВ CI/CD-RUNNER НА БАЗЕ ГИБРИДНОГО АНАЛИЗА УГРОЗ

Николай В. Корнеев[1]✉, Дмитрий И. Шиллер[2,3]
[1]РГУ нефти и газа (НИУ) им. И.М. Губкина, Ленинский пр–кт, 65, Москва, 119991, Россия
[2]Финансовый университет при Правительстве Российской Федерации, Ленинградский пр–кт, 49/2, Москва, 125167, Россия
[3]ООО «Кросстех Солюшнс Групп», вн. тер. г. муниципальный округ Солнцево, Киевское ш., 22-й км, 6А, стр. 1, Москва, 108811,Россия
✉niccyper@mail.ru

Аннотация. Целью исследования является предотвращение несанкционированного доступа к среде сборки и утечки конфиденциальных данных в процессе непрерывной интеграции путем разработки и обоснования архитектурного паттерна обеспечения безопасности специального агентского сервиса CI/CD-runner (Continuous Integration/Continuous Delivery-runner). Методология проведения исследования включала в себя использование методов обработки с опорой, как на сигнатурные правила, так и на алгоритмы выявления аномалий, применение специализированных механизмов защиты в сочетании с превентивными мерами, а также архитектурное разграничение функций. В ходе исследования обоснован архитектурный паттерн обеспечения безопасности специального агентского сервиса CI/CD-runner, который позволит предотвратить несанкционированный доступ к среде сборки и утечку конфиденциальных данных в процессе непрерывной интеграции. Проведен анализ угроз безопасности и существующих подходов к защите CI/CD, предложены классификация актуальных угроз безопасности для CI/CD-runner и защитные механизмы с учетом их влияния на показатели CI/CD, каталог паттернов безопасности для CI/CD-runner. Сформулированы принципы построения паттернов безопасности для агентских сервисов CI/CD-runner на базе концепции гибридного анализа угроз и разработана унифицированная схема модели их построения в нотации UML. Новизна полученных решений включает методологические принципы построения паттернов безопасности для агентских сервисов CI/CD-runner на базе концепции гибридного анализа угроз, которые отсутствуют, как в зарубежных, так и в российских научных исследованиях, что позволит разрабатывать новые архитектурные механизмы, повышающие безопасность CI/CD-runner без уменьшения их производительности, и тем самым минимизировать риски компрометации среды выполнения задач CI/CD. Практическая ценность определяется тем, что исследование проведено в сегменте разработки программного обеспечения и системной интеграции. Безопасность конвейеров CI/CD является фундаментальной проблемой для всех компаний, занимающихся созданием цифровых продуктов, однако наиболее остро она стоит перед организациями, работающими с критической информационной инфраструктурой или обрабатывающими персональные данные.

Ключевые слова: *непрерывная интеграция, непрерывная доставка, пайплайн, агент для выполнения заданий из CI/CD-пайплайнов, несанкционированный доступ, утечка конфиденциальных данных, шаблон, гибридный анализ угроз, политика безопасности*

Для цитирования: Корнеев, Н., Шиллер, Д. (2026). Принципы построения паттернов безопасности для агентских сервисов CI/CD-runner на базе гибридного анализа угроз. *Безопасность информационных технологий*, 33(4), 143-156. doi: http://dx.doi.org/10.26583/bit.2026.4.12

PRINCIPLES FOR CONSTRUCTING SECURITY PATTERNS FOR CI/CD-RUNNER AGENT SERVICES BASED ON HYBRID THREAT ANALYSIS

Nikolay V. Korneev[1]✉, Dmitry I. Shiller[2,3]
[1]Gubkin Russian State University of Oil and Gas (National Research University), Leninsky Prospect, 65, Moscow, 119991, Russia
[2]Financial University under the Government of the Russian Federation, Leningradsky Prospect, 49/2, Moscow, 125167, Russia

[3]*Crosstech Solutions Group LLS, 6A Bldg. 1, 22nd km Kievskoe sh., Solntsevo, Moscow, 108811, Russia*
✉*niccyper@mail.ru*

Abstract. The aim of this study is to prevent unauthorized access to the build environment and leakage of confidential data during continuous integration by developing and substantiating an architectural security pattern for the CI/CD-runner (Continuous Integration/Continuous Delivery-runner) agent service. The research methodology included processing methods based on both signature rules and anomaly detection algorithms, as well as specialized protection mechanisms combined with preventive measures and architectural separation of functions. As a result of the research, the architectural security pattern for the CI/CD-runner agent service has been substantiated to prevent unauthorized access to the build environment and leakage of confidential data during continuous integration. The study analyzed security threats and existing approaches to CI/CD protection, and proposed a classification of current security threats for CI/CD-runner and protective mechanisms that account for their impact on CI/CD performance, along with a catalog of security patterns for CI/CD-runner. Additionally, the principles for constructing security patterns for CI/CD-runner agent services based on the concept of hybrid threat analysis have been formulated and a unified UML model scheme for their construction has been developed. The novelty of the proposed solutions includes methodological principles for constructing security patterns for CI/CD-runner agent services based on hybrid threat analysis, which are currently absent in both foreign and Russian scientific research. These principles will enable the development of new architectural mechanisms that enhance CI/CD-runner security without reducing their performance, thus minimizing the risks of compromising the CI/CD task execution environment. The practical value of the work is determined by the fact that the research was conducted in the software development and system integration segment. CI/CD pipeline security is a fundamental issue for all companies developing digital products, but it is particularly acute for organizations handling critical information infrastructure or processing personal data.

Keywords: *continuous integration, continuous delivery, pipeline, CI/CD-runner, unauthorized access, data leakage, security pattern, hybrid threat analysis, policy engine*

For citation: Korneev, N., Shiller, D. (2026). Principles for constructing security patterns for CI/CD-runner agent services based on hybrid threat analysis. *IT Security (Russia)*, 33(4), 143-156. doi: http://dx.doi.org/10.26583/bit.2026.4.12

Введение

Непрерывная интеграция и доставка программного обеспечения стали очень популярны в настоящее время как у зарубежных, так и у российских разработчиков, в то время как широкое распространение систем автоматизации разработки привело к формированию новых видов атак. Одним из уязвимых элементов является CI/CD-runner (Continuous Integration /Continuous Delivery-runner) – агент, выполняющий инструкции пайплайна [1–3]. Несмотря на активную интеграцию DevOps-практик [3], существующие решения ограничиваются разрешёнными образами контейнеров и политиками безопасности, что не исключает интерпретацию корректных на вид команд как вредоносных. Таким образом, требуется разработка архитектурного механизма, повышающего безопасность CI/CD-runner без уменьшения их производительности.

Однако решение такой масштабной задачи невозможно без определения методологических принципов построения такого архитектурного механизма на базе паттернов безопасности [4, 5]. Это особенно актуально, в связи с тем, что существует пробел в стандартизации паттерна безопасности для специальных агентских сервисов. Существующие решения, например [6] часто требуют сложной интеграции или не являются гибкими для специфических требований российской организации-партнера. Существующие подходы, например для Machine Learning Operations (MLOps) [7] разрозненны: одни работают до запуска, другие во время, но нет целостного паттерна, который бы сочетал превентивный анализ по правилам и поведенческий анализ на базе машинного обучения (Machine Learning) непосредственно на уровне runner, что и обуславливает актуальность данной статьи.

Целью исследования является предотвращение несанкционированного доступа к среде сборки и утечки конфиденциальных данных в процессе непрерывной интеграции путем разработки и обоснования архитектурного паттерна обеспечения безопасности специального агентского сервиса CI/CD-runner. Он позволит предотвратить

несанкционированный доступ к среде сборки и утечку конфиденциальных данных в процессе непрерывной интеграции.

Для достижения цели в данной статье на первом этапе проведем анализ угроз безопасности и существующих подходов к защите CI/CD, предложим классификацию актуальных угроз безопасности для CI/CD-runner и защитные механизмы с учетом их влияния на показатели CI/CD, определим каталог паттернов безопасности для CI/CD-runner, сформулируем принципы построения паттернов безопасности для агентских сервисов CI/CD-runner на базе концепции гибридного анализа угроз и разработаем унифицированную схему модели их построения в нотации UML (Unified Modeling Language).

Практическая ценность определяется тем, что исследование ведется в интересах отрасли информационных технологий и связи, а именно в сегменте разработки программного обеспечения и системной интеграции. Безопасность конвейеров CI/CD является фундаментальной проблемой для всех компаний, занимающихся созданием цифровых продуктов, однако наиболее остро она стоит перед организациями, работающими с критической информационной инфраструктурой или обрабатывающими персональные данные.

Прикладное значение исследования реализуется в интересах российской организации ООО «Кросстех Солюшнс Групп». Выбор данной организации обусловлен её специализацией на разработке программного обеспечения для информационной безопасности. Компания «Кросстех Солюшнс Групп» создает продукты, которые используются для защиты информационных активов других предприятий. В связи с этим, требования к безопасности внутренней инфраструктуры разработки в данной компании значительно выше, чем в среднем по рынку.

1. Анализ угроз безопасности и существующих подходов к защите CI/CD

Под CI/CD в настоящей работе понимается совокупность процессов непрерывной интеграции, доставки и развертывания, обеспечивающих регулярное включение изменений в общую кодовую базу, их автоматическую проверку и подготовку к выпуску. CI/CD не сводится только к автоматическому деплою, а формированию воспроизводимого технологического контура, в котором каждое изменение проходит единообразные этапы сборки, тестирования и контроля качества [8, 9].

Непрерывная интеграция ориентирована на частое объединение изменений и раннее выявление дефектов совместимости, тогда как непрерывная поставка обеспечивает автоматизированную подготовку программного продукта к выпуску, а непрерывное развертывание доводит автоматизацию до уровня прямого переноса проверенных изменений в эксплуатационную среду. Для корпоративной разработки это означает переход от эпизодических релизов к непрерывному конвейеру инженерных операций, где качество, воспроизводимость и безопасность должны обеспечиваться на каждом шаге [10, 11].

Рост зрелости CI/CD изменяет и сам объект защиты. Если в классической схеме основное внимание уделялось коду и продуктивной среде, то в DevSecOps-подходе необходимо защищать также правила оркестрации, описание этапов сборки, механизмы получения зависимостей, хранилища артефактов и учетные данные, используемые внутри пайплайна. Следовательно, компрометация конвейера может приводить к внедрению недоверенного кода еще до развертывания приложения у конечного потребителя [11, 12].

Для задач информационной безопасности CI/CD следует рассматривать как часть программной цепочки поставки. В таком понимании объектом защиты становятся не только исходные тексты и продуктивная среда, но и сами конфигурации пайплайна, внешние зависимости, контейнерные образы, артефакты сборки, служебные токены и сценарии автоматизации [13, 14]. Следовательно, компрометация конвейера может приводить к распространению недоверенного кода уже на этапе сборки, а не только на этапе эксплуатации.

CI/CD-runner представляет собой программный агент, регистрируемый в системе управления пайплайнами и предназначенный для фактического исполнения jobs. Согласно официальной документации GitLab Runner, агент ожидает задания от CI/CD-платформы,

получает параметры job, подготавливает среду и возвращает результаты выполнения обратно в систему.

В прикладном смысле runner является исполнительным узлом автоматизации. Через него проходят операции извлечения исходного кода, подготовки рабочего каталога, подстановки переменных окружения, запуска shell-команд, сборки контейнерных образов, публикации артефактов и деплоя в целевые среды. Таким образом, runner совмещает в себе признаки сервисного агента, изолированной среды исполнения и точки доступа к критическим ресурсам разработки[1].

Высокая чувствительность runner-агента определяется тем, что он одновременно интерпретирует пользовательские сценарии, обладает правами на выполнение команд и способен инициировать действия в инфраструктуре компании. При его компрометации возникает риск не только локального сбоя сборки, но и подмены артефактов, распространения вредоносной логики по цепочке поставки, получения доступа к внутренним репозиториям и нарушения целостности релизного процесса[2].

Назначение runner-агента состоит в разделении логики оркестрации и логики вычислительного исполнения. Платформа CI/CD определяет, какие этапы и в какой последовательности должны выполняться, тогда как runner реализует запуск команд в заданном окружении – например, в shell, контейнере или иной среде[3]. Именно поэтому runner должен рассматриваться как критически важный объект защиты, так как он получает доступ к исходному коду, временным данным сборки, сетевым ресурсам и секретам, необходимым для выпуска программного продукта[4].

Принцип работы runner-агента можно описать как последовательность стадий (рис. 1), сначала происходит регистрация в системе CI/CD, после ожидание задания, получение job, подготовка среды, выполнение сценария и возврат результатов. В документации GitLab также подчеркивается, что выбор конкретного runner осуществляется по признакам соответствия тегам, разрешениям и доступности ресурса.

С архитектурной точки зрения существенное значение имеет режим изоляции, в котором работает runner. Чем ближе среда исполнения к хостовой операционной системе и чем шире ее привилегии, тем выше риск горизонтального распространения атаки при выполнении недоверенного сценария. Поэтому принцип работы runner необходимо анализировать не только как функциональную последовательность этапов, но и как систему доверенных границ между платформой CI/CD, средой исполнения и корпоративной инфраструктурой.

Дополнительную проблему создает тот факт, что runner исполняет сценарии, подготовленные различными участниками процесса разработки, включая разработчиков, DevOps-инженеров и авторов процесса объединения кода (merge/pull request). Следовательно, код, поступающий в среду исполнения, не может априори считаться полностью доверенным. Из этого вытекает требование к специальным защитным механизмам, способным оценивать не только корректность конфигурации, но и ее потенциальную опасность до и во время выполнения [13].

На уровне отдельного job процесс включает получение исходных данных, подготовку окружения, выполнение пользовательских команд и выгрузку результатов. GitLab описывает выполнение job как набор последовательных шагов, что особенно важно для анализа безопасности. Любая команда, попавшая в сценарий выполнения, получает возможность влиять на файловую систему, процессы и сетевое окружение runner-узла.

Одной из наиболее опасных категорий угроз для runner-агентов являются инъекционные атаки в CI/CD-скриптах. Их сущность состоит в том, что недоверенный ввод или специально сформированная строка параметров интерпретируются системой исполнения как команда. Необходимо раннее выявление опасных сценариев автоматизации, поскольку вредоносная логика может быть скрыта уже в конфигурации пайплайна или подключаемом скрипте [13, 14].

[1]GitLab. GitLab Runner. URL: https://docs.gitlab.com/runner/ (дата обращения: 06.04.2026).
[2]GitLab. Job execution flow. URL: https://docs.gitlab.com/ci/jobs/job_execution/ (дата обращения: 06.04.2026).
[3]GitLab. Configuring runners. URL: https://docs.gitlab.com/ci/runners/configure_runners/ (дата обращения: 07.04.2026).
[4]GitLab. Security for self-managed runners. URL: https://docs.gitlab.com/runner/security/ (дата обращения: 06.04.2026).



Рис. 1. UML-диаграмма процесса работы runner-агента

В прикладном плане инъекционные атаки в CI/CD могут проявляться в трех типичных формах. Во-первых, это командные инъекции в shell-скриптах, когда значения переменных, имена веток, сообщения коммитов или параметры сборки подставляются в командную строку без экранирования. Во-вторых, это инъекции в языках шаблонизации и сборочной логики, где вредоносный фрагмент маскируется под часть YAML-конфигурации или скрипта генерации job. В-третьих, это инъекции через внешние зависимости и установочные сценарии, когда пайплайн автоматически получает и исполняет недоверенный код из внешнего источника [15].

Опасность данного класса атак определяется тем, что исполнение происходит внутри доверенной автоматизированной среды, обладающей повышенными полномочиями и доступом к чувствительным данным. В результате успешная инъекция может обеспечить злоумышленнику чтение секретов, изменение артефактов, установление сетевого канала управления или закрепление во внутреннем контуре организации под видом штатной операции сборки.

Для GitHub Actions риск описан на уровне официальной документации, злоумышленник способен встроить вредоносные команды в контекст, а затем они будут выполнены на runner, как часть штатного процесса[5]. Аналогичный риск существует и в self-hosted конвейерах, где в качестве недоверенного ввода могут выступать имена веток, комментарии к изменениям в коде, параметры сборки, значения переменных и содержимое конфигурационных файлов.

Самостоятельную группу угроз образуют команды, перенаправляющие выполнение или сетевое взаимодействие на внешние подозрительные источники. Речь идет о скачивании скриптов и бинарных файлов из несанкционированных доменов, обращении к недоверенным

[5]GitHub Docs. Script injections. URL: https://docs.github.com/en/actions/concepts/security/script-injections (дата обращения: 06.04.2026).

registry и пакетным репозиториям, а также о передаче результатов job на внешние узлы. Одна из центральных проблем CI/CD связана с зависимостями и внешними источниками артефактов, через которые в конвейер могут попадать вредоносные компоненты [15, 16].

Наиболее рискованными являются сценарии типа «загрузить и немедленно выполнить», при которых runner обращается к удаленному ресурсу, получает скрипт или бинарный файл и сразу передает его интерпретатору. С точки зрения внешнего наблюдателя такие операции могут выглядеть как обычная установка зависимости или инструмента сборки, однако фактически создают канал доставки недоверенного исполняемого содержимого внутрь доверенной среды.

Для корпоративной инфраструктуры важно различать допустимые обращения к доверенным репозиториям и registry, сомнительные обращения к несанкционированным зеркалам и явно вредоносные конструкции, ориентированные на скрытое скачивание кода либо эксфильтрацию данных. В российских ИТ-компаниях, использующих внутренние пакетные зеркала и локальные цепочки поставки, контроль источников становится не только технической, но и организационной мерой обеспечения импортонезависимости и технологической управляемости [11].

Для инфраструктуры, содержащей runner-агентов, опасность усугубляется тем, что подобные обращения нередко выполняются автоматически и не вызывают подозрения у разработчика, если внешняя команда встроена в сценарий сборки. Поэтому контроль допустимых источников загрузки и сетевых направлений должен рассматриваться как самостоятельный элемент защиты пайплайна, а не как второстепенная настройка сетевого экрана. Особенно это важно для самоуправляемых runner-агентов (self-managed runners), где выполнение недоверенных команд происходит во внутреннем контуре организации.

Утечки секретов относятся к наиболее критичным последствиям компрометации CI/CD. Для runner-агента секретами выступают токены доступа, SSH-ключи, параметры доступа к облачным сервисам, переменные окружения, а также служебные токены самой инфраструктуры, содержащей runner-агентов. Конфиденциальные данные должны контролироваться уже на ранних этапах разработки и сопровождаться механизмами выявления их несанкционированного использования [11, 17].

Каналы утечки секретов в CI/CD достаточно разнообразны. Чувствительные данные могут появляться в логах выполнения, во временных файлах, в артефактах сборки, в кэше зависимостей, в аргументах командной строки и в сетевых запросах к внешним сервисам. Дополнительный риск создают ошибки в разграничении прав, когда один job или контейнер получает доступ к учетным данным, предназначенным для другого этапа пайплайна.

Критичность утечек определяется тем, что последствия выходят далеко за пределы конкретной сборки. Один компрометированный токен может дать возможность изменить код, опубликовать подмененный артефакт, зарегистрировать ложный runner, инициировать несанкционированный деплой или закрепиться во внутренней инфраструктуре. Поэтому защита секретов в CI/CD-runner должна строиться вокруг принципов минимально необходимого доступа, короткого времени жизни учетных данных, маскирования значений в логах.

С технической точки зрения утечка может происходить через вывод переменных в логи, чтение секретов из окружения исполняемым процессом, передачу данных во внешний HTTP запрос или публикацию чувствительных данных в артефактах сборки. В документации GitLab отдельно подчеркивается важность защиты токенов и ограничения прав runner-агентов, поскольку компрометация служебного токена позволяет злоумышленнику злоупотреблять инфраструктурой выполнения[6].

Таким образом можно представить классификацию актуальных угроз безопасности для CI/CD-runner в форме табл. 1. В табл. 1 сопоставлены ключевые угрозы и базовые контрольные показатели, которые могут быть использованы для оценки новых архитектурных механизмов, повышающих безопасность CI/CD-runner.

[6]GitLab token overview. URL: https://docs.gitlab.com/security/tokens/ (дата обращения: 06.04.2026).

Таблица 1. Классификация актуальных угроз безопасности для CI/CD-runner

Тип угрозы	Основное проявление	Контролируемые метрики	Базовый диапазон для оценки
Инъекционные атаки	Выполнение несанкционированных команд, интерпретация недоверенного ввода	Точность выявления, полнота выявления	Точность 0.60-0.75, полнота 0.55-0.70
Подозрительные внешние источники	Скачивание и запуск внешнего содержимого, обращения к недоверенным доменам	Точность выявления, полнота выявления	Точность 0.65-0.80, полнота 0.60-0.75
Утечки секретов	Вывод токенов, чтение чувствительных переменных, публикация в логах и артефактах	Точность выявления, полнота выявления, доля ложных срабатываний	Точность 0.60-0.75, полнота 0.70-0.90, доля ложных срабатываний 0.0-0.10

В табл. 1 представлены не итоговые экспериментальные значения, а контрольные интервалы, используемые в работе для последующего сопоставления с результатами после внедрения паттернов безопасности.

К числу базовых подходов защиты относится статический анализ CI/CD-скриптов, выполняемый до запуска job. Его цель состоит в обнаружении опасных конструкций на этапе описания пайплайна, например, обращений к внешним источникам, использования чувствительных данных в явном виде [13, 14].

Статический анализ в контексте CI/CD включает не только поиск явно вредоносных команд, он также охватывает обнаружение небезопасных конструкций, обращений к недоверенным источникам, ошибок в обращении с секретами, неограниченного сетевого доступа. Тем самым статический контроль превращается в механизм ранней фильтрации сценариев, еще не получивших доступа к вычислительной среде runner [13, 17].

Каждое срабатывание можно сопоставить с конкретным правилом, зафиксировать в отчете и использовать как основание для блокировки или ручной проверки. Ограничение состоит в том, что сложные цепочки команд могут не попасть под заранее определенные сигнатуры. Поэтому статический анализ является необходимым, но не достаточным слоем защиты [14].

Преимущество статического анализа состоит в переносе контроля на раннюю стадию жизненного цикла. Опасный сценарий может быть выявлен еще до запуска job на runner, то есть до предоставления вредоносному коду доступа к вычислительным ресурсам и секретам. Вместе с тем данный анализ ограничен заранее описанными шаблонами и требует дополнения поведенческим контролем, способным выявлять ранее неизвестные сценарии злоупотребления допустимыми командами [11].

Вторым важным направлением защиты являются политики безопасности, задающие допустимые правила работы пайплайн и runner-агентов.

Политики безопасности (Policy Engine) в CI/CD задают рамку допустимого поведения для проектов, веток, типов окружений и классов runner-агентов. Внутри такой рамки формализуются перечни разрешенных команд и интерпретаторов, допустимые сетевые направления, условия использования привилегированных контейнеров, порядок выдачи секретов, правила публикации артефактов и критерии блокировки задания. За счет этого защита перестает быть реакцией на отдельные инциденты и становится воспроизводимым механизмом управления рисками [11, 17].

Для runner-инфраструктуры особенно важно связывать policy-ограничения с конкретным уровнем доверия к заданию. Например, чувствительные jobs могут исполняться только на выделенных runner-узлах и только для защищенных веток, тогда как менее доверенные сценарии запускаются в более ограниченной среде. Такой подход уменьшает вероятность того, что опасный сценарий получит доступ к расширенным ресурсам организации лишь потому, что формально соответствует синтаксису конфигурации GitLab.

Для runner-инфраструктуры политики безопасности включают ограничение областей применения конкретных runner-агентов, разделение доверенных и недоверенных задач, привязку чувствительных jobs к защищенным веткам и тегам, а также регламентацию допустимых источников загрузки и сетевых взаимодействий. Документация GitLab позволяет технически реализовывать такие ограничения через protected runners, теги и настройки области видимости job, что делает policy-механизмы исполнимыми на практике.

Третье направление связано с динамическим анализом поведения runner-агента во время исполнения job. В этом случае объектом контроля становятся уже не YAML-конфигурации, а реальные процессы, файловые операции, обращения к сети, последовательности команд и взаимодействие с секретами. В работах по созданию систем мониторинга инцидентов информационной безопасности подчеркивается ценность раннего и непрерывного сбора событий, встроенного в технологический цикл разработки [17].

Практическая ценность динамического анализа состоит в наблюдении за тем, что задание реально делает в среде выполнения. Объектами контроля становятся запускаемые процессы и их аргументы, дочерние процессы, файловые операции, сетевые соединения, обращения к переменным окружения и секретам, а также временной профиль работы job. Такой подход позволяет фиксировать не потенциальные намерения, а фактическое поведение, включая вредоносные действия, сформированные уже после старта исполнения [18].

Особое значение динамический контроль имеет в случаях, когда безопасно выглядящая команда инициирует нетипичное сетевое обращение, скачивает внешний скрипт, создает цепочку дочерних процессов или начинает считывать чувствительные данные, нехарактерные для данного класса job. Именно поэтому телеметрия времени выполнения должна рассматриваться как обязательное дополнение к статическому анализу и контролю.

С точки зрения методов обработки такой подход может опираться как на сигнатурные правила, так и на алгоритмы выявления аномалий. Аномальное поведение проявляется через отклонения от типового профиля трафика и последовательности событий, а потому хорошо подходит для выявления неизвестных или слабо формализуемых атак [18]. Для runner-инфраструктуры это означает возможность обнаруживать нетипичные вызовы команд, неожиданные исходящие соединения и аномальный доступ к чувствительным данным уже в момент выполнения пайплайна.

Таким образом обобщая подходы и направления защиты среды выполнения задач CI/CD можно выделить защитные механизмы приведенные в табл. 2.

Таблица 2. Сравнение защитных механизмов и их влияния на показатели CI/CD

Механизм	Объект контроля	Критичные правила, политики	Влияние на метрики
Статический анализ	YAML-конфигурации, сигнатуры опасных команд	Запрет опасных шаблонов, контроль недоверенных параметров, проверка обращения к секретам	Рост точности выявления известных угроз
Политики безопасности	Доступ runner, ветки, теги, привилегии контейнеров, допустимые источники	Белые списки источников, разграничение доверенных и недоверенных job	Снижение площади атаки
Динамический анализ	Процессы, сеть, файловые операции, обращения к секретам, поведенческий профиль	Мониторинг аномалий, контроль отклонения от нормального профиля	Снижение риска исполнения недоверенных сценариев

В табл. 2 выделены ключевые защитные механизмы и их влияния на показатели CI/CD, через объекты контроля, критичные правила, политики и метрики.

2. Каталог паттернов безопасности для CI/CD-runner

В контексте настоящей работы под паттерном понимается повторно применимый шаблон решения типовой задачи безопасности в определенном архитектурном контексте. Паттерны рассматриваются как универсальные шаблонные механизмы защиты, позволяющие формализовать типовую угрозу, выделить архитектурные компоненты защиты и воспроизводимо применять решение в различных системах [4].

Как инженерная конструкция паттерн обычно включает описание контекста применения, типовой проблемы, набора структурных элементов, алгоритма противодействия угрозе и ожидаемого защитного эффекта. Такой формат позволяет не просто зафиксировать полезную рекомендацию, а превратить ее в воспроизводимый механизм проектирования и внедрения защитных средств внутри конкретной архитектуры [5].

Для темы CI/CD-runner паттерн особенно ценен тем, что переносит обсуждение безопасности из плоскости частных настроек в плоскость повторно применимых решений. Один и тот же паттерн может использоваться для разных проектов организации, различаясь параметрами политик, правилами анализа и допустимыми профилями поведения, но сохраняя единую логическую структуру защиты.

Для исследования такой подход особенно важен, поскольку переводит защиту CI/CD-runner из набора разрозненных рекомендаций в формат инженерно-воспроизводимого решения. Паттерн описывает контекст применения, тип угрозы, структурные элементы, механизм работы и ожидаемый защитный эффект. Это делает возможной разработку специального агентского сервиса, который можно тиражировать между проектами одной организации [4, 5].

Первый паттерн предназначен для предотвращения запуска опасных или потенциально вредоносных команд до фактического исполнения job на runner. Его сущность состоит в предварительном разборе CI/CD-скрипта, выделении исполняемых команд и сопоставлении их с набором запрещенных или высокорисковых признаков таких как использование интерпретируемых shell-конструкций, передача недоверенного ввода в командную строку и иных опасных шаблонов.

Логика паттерна включает несколько этапов. Первый этап включает в себя это разбор конфигурации пайплайн. После первого этапа происходит извлечение командного содержимого, нормализация строк, проверка на сигнатуры опасных конструкций и формирование решения о пропуске, ограничении или блокировке job. В расширенном варианте паттерн может учитывать не только отдельные команды, но и их композиции, например передачу вывода в shell-интерпретатор и подстановку недоверенного ввода.

С практической точки зрения данный паттерн обеспечивает наиболее ранний контур защиты runner-агента. Он позволяет остановить вредоносный или сомнительный сценарий еще до предоставления ему вычислительных ресурсов и доступа к секретам. В то же время эффективность паттерна зависит от полноты базы признаков и от способности системы различать действительно опасные комбинации и допустимые служебные команды, применяемые в легитимных целях [4].

С инженерной точки зрения такой паттерн должен включать парсер конфигурации пайплайн, нормализацию и лексический разбор команд, rule-based движок и механизм принятия решения о блокировке или переводе job на дополнительное согласование. Концептуально он близок к паттернам, где защита строится как повторно применимая микросервисная схема вокруг типовой угрозы, а сам защитный механизм отделяется от прикладной бизнес-логики [5].

Второй паттерн ориентирован на контроль внешних источников, с которыми взаимодействует runner-агент. Основная идея заключается в формировании белых списков

доменов, IP-адресов, package registry, контейнерных registry и внутренних репозиториев, разрешенных для конкретного класса пайплайна. Все обращения за пределы этого множества должны либо блокироваться, либо выноситься на дополнительную проверку [16].

С инженерной точки зрения паттерн белых списков источников должен быть контекстно-зависимым. Для разных проектов, стадий пайплайна и сред разработки допустимыми могут быть разные наборы доменов, registry и внутренних сервисов. Поэтому эффективная реализация предполагает не один глобальный список, а иерархию правил, привязанных к типу job, уровню доверия к runner и корпоративным сетевым ограничениям.

Поддержка такого паттерна требует постоянной актуализации перечня доверенных источников и журналирования отклонений. Однако именно эта мера резко снижает вероятность скрытого скачивания вредоносного кода, подмены зависимости на внешнем зеркале и эксфильтрации данных через неожиданные сетевые каналы. Для self-managed runners, работающих во внутреннем контуре организации, она является одним из наиболее практичных и управляемых способов ограничения поверхности атаки.

Необходимость такого паттерна обусловлена тем, что вредоносная активность в CI/CD часто маскируется под легитимные операции скачивания зависимостей, образов и скриптов. В результате опасной становится не только явно вредоносная команда, но и формально допустимый вызов штатной утилиты, направленный на внешний недоверенный ресурс. Для self-managed runners, размещенных во внутреннем контуре компании, контроль направлений сетевого взаимодействия становится обязательным слоем защиты.

Третий паттерн направлен на защиту секретов и служебных токенов в runner-инфраструктуре. Он строится на принципах минимизации прав, кратковременности действия учетных данных, исключения секретов из открытого текста CI/CD-конфигураций, маскирования чувствительной информации в логах и контроля фактов обращения к секретам в момент выполнения job [11, 17].

В жизненном цикле данного паттерна необходимо выделять стадии хранения, выдачи, использования, журналирования и удаления чувствительных данных. На каждой из этих стадий должны действовать собственные ограничения. Необходимо централизованное хранение секретов, выдача только на время их исполнения, привязка к роли или конкретному этапу пайплайна, запрет вывода в логи и очистка временного окружения после завершения job.

Важной особенностью является сочетание превентивных и наблюдательных механизмов. Даже если секрет не хранится в открытом виде, система должна обнаруживать признаки его опасного использования, например вывод в stdout. В этом смысле паттерн защиты секретов напрямую связан с общей концепцией гибридного анализа, поскольку опирается и на правила, и на поведенческое наблюдение [5].

В методологическом плане данный паттерн также может быть описан как специализированный шаблонный механизм защиты. Паттерн становится особенно эффективным, когда сочетает превентивные меры, архитектурное разграничение функций и последующий мониторинг событий безопасности [5]. Для CI/CD-runner это означает необходимость не только безопасного хранения секретов, но и анализа попыток их чтения, вывода и передачи во внешние каналы.

3. Концепция гибридного анализа угроз

Предлагаемая концепция гибридного анализа угроз включает два слоя: статический анализ по правилам (первый слой), машинное обучение для выявления аномалий (второй слой) и совмещение слоев в виде гибридной модели. На рис. 2 показана UML-диаграмма процесса работы CI/CD-runner с интегрированными паттернами безопасности на базе концепции гибридного анализа угроз.

Первый слой предлагаемой защиты должен строиться на статическом анализе по правилам (рис. 2). Он обеспечивает раннее выявление нарушений, ввода опасных команд,

недопустимых сетевых обращений, включения чувствительных данных в пайплайн-конфигурацию.

В методологическом плане анализ должен опираться на формализованную базу признаков, отражающую как общие практики безопасной разработки, так и специфику конкретной организации. В такую базу могут входить сигнатуры вредоносных команд, шаблоны подозрительных сетевых вызовов, ограничения на использование привилегированных режимов, правила обращения с секретами и критерии допустимости для разных классов runner-агентов. Чем точнее эта база привязана к реальному CI/CD-процессу компании, тем выше ее практическая ценность.

Статические правила удобны своей объяснимостью и низкой стоимостью проверки. Они позволяют обосновывать решение о блокировке пайплайна, формировать отчетность и тиражировать единые стандарты между командами. Однако одних правил недостаточно, так как злоумышленник может использовать новые формы обфускации, легитимные утилиты или сложные многошаговые последовательности действий, которые не будут прямо описаны в сигнатурной базе.

Второй слой защиты целесообразно строить на выявлении аномалий в поведении runner-агента. В этом случае модель (рис. 2) анализирует не только отдельную команду, но и совокупность признаков, например: последовательность запускаемых процессов, характер сетевых соединений, длительность стадий job, частоту обращений к секретам, профиль файловых операций и отклонения от типового поведения конкретного класса пайплайна.

Для обучения моделей аномалий может использоваться частота использования команд, порядок их следования, длительность стадий job, интенсивность обращений к секретам и отклонение проекта. Такой набор признаков позволяет анализировать не только отдельные события, но и поведенческий контекст, в котором они возникают.

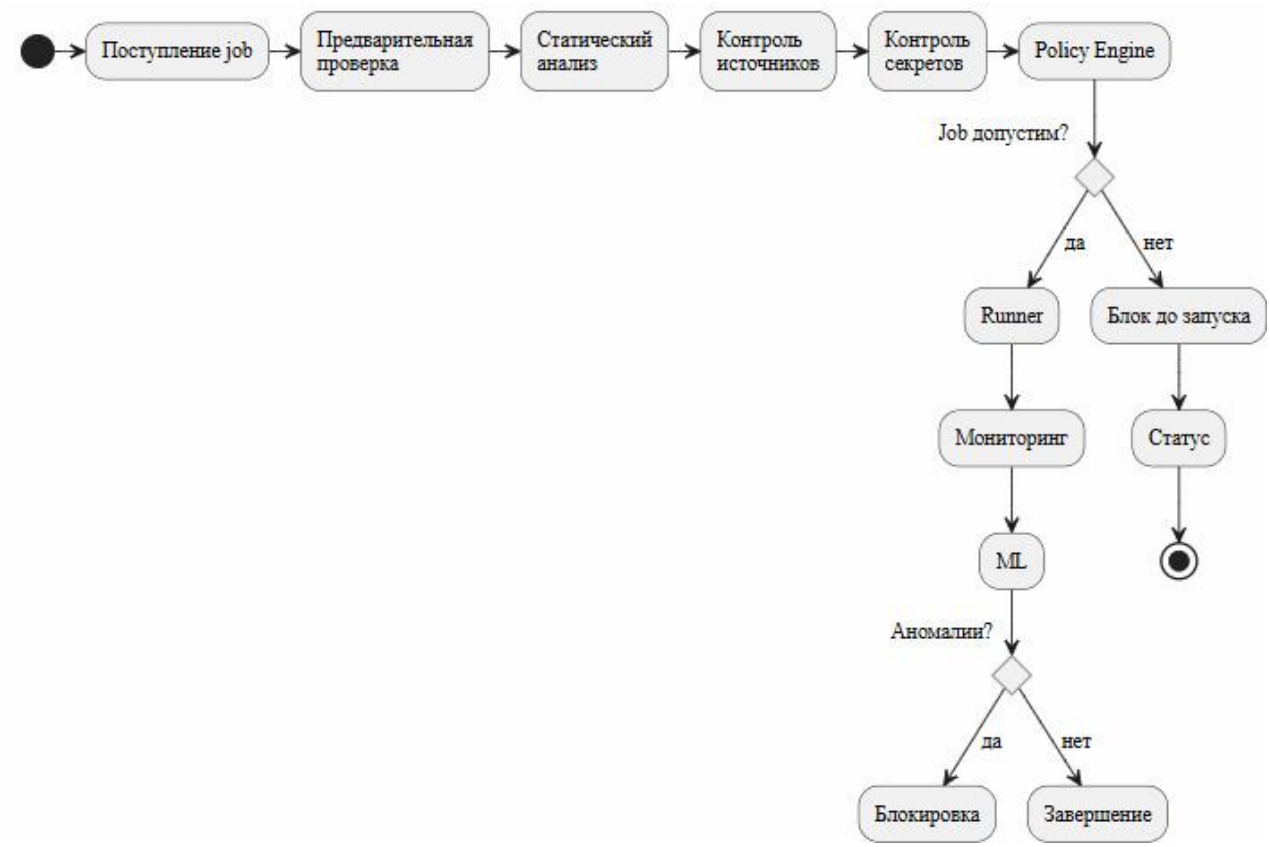


Рис. 2. UML-диаграмма процесса работы CI/CD-runner с интегрированными паттернами безопасности на базе концепции гибридного анализа угроз

Преимущество подхода состоит в способности обнаруживать новые или слабо формализуемые сценарии, в том числе случаи, когда каждая отдельная команда выглядит допустимой, но вся последовательность действий нетипична для данного класса пайплайнов. Ограничением остаются требования к качеству исторических данных, объяснимости результатов и устойчивости модели к изменению нормального поведения при развитии CI/CD практик.

Преимущество данного подхода состоит в способности обнаруживать ранее неизвестные или трудно формализуемые сценарии. Даже если отдельная команда не находится в черном списке, система может распознать, что текущий job выполняет нетипичную последовательность действий или обращается к необычному источнику. Ограничением является зависимость от качества обучающей выборки и риска ложных срабатываний (рис. 2), поэтому модели машинного обучения (ML) должны использоваться не изолированно, а совместно с правилами и политиками безопасности (Policy Engine).

Наиболее перспективной для защиты CI/CD-runner является гибридная модель, в которой статический анализ и поведенческое выявление аномалий работают совместно (рис. 2). Статический уровень блокирует известные опасные сценарии еще до запуска job, а динамический уровень контролирует фактическое исполнение и фиксирует отклонения, которые невозможно полностью описать заранее.

С точки зрения архитектуры специального агентского сервиса это означает наличие двух контуров контроля. До запуска задания сервис анализирует пайплайн-конфигурацию, формирует предварительное решение и определяет допустимый профиль поведения. Во время выполнения job сервис собирает телеметрию процессов, команд, сетевых соединений и обращений к секретам, после завершения выполнения агрегирует результаты для совершенствования правил и моделей. Такой подход соответствует современной логике DevSecOps и позволяет адаптировать защиту к специфике конкретной российской ИТ-компании.

Заключение

В ходе исследования обоснован архитектурный паттерн обеспечения безопасности специального агентского сервиса CI/CD-runner, который позволит предотвратить несанкционированный доступ к среде сборки и утечку конфиденциальных данных в процессе непрерывной интеграции. Проведен анализ угроз безопасности и существующих подходов к защите CI/CD, предложены классификация актуальных угроз безопасности для CI/CD-runner и защитные механизмы с учетом их влияния на показатели CI/CD, каталог паттернов безопасности для CI/CD-runner. Сформулированы принципы построения паттернов безопасности для агентских сервисов CI/CD-runner на базе концепции гибридного анализа угроз и разработана унифицированная схема модели их построения в нотации UML.

Новизна полученных решений включает методологические принципы построения паттернов безопасности для агентских сервисов CI/CD-runner на базе концепции гибридного анализа угроз, которые отсутствуют, как в зарубежных, так и в российских научных исследованиях, что позволит разрабатывать новые архитектурные механизмы, повышающие безопасность CI/CD-runner без уменьшения их производительности, и тем самым минимизировать риски компрометации среды выполнения задач CI/CD.

СПИСОК ЛИТЕРАТУРЫ/REFERENCES:

1. Muhammad Waseem, Peng Liang, Aakash Ahmad, Arif Ali Khan, Mojtaba Shahin, Ali Rezaei Nasab, Tommi Mikkonen, Pekka Abrahamsson. Understanding the issues, their causes and solutions in microservices systems: An empirical study. Journal of Systems and Software. V. 237, 2026, 112828. DOI: https://doi.org/10.1016/j.jss.2026.112828.
2. Anisha Kumari, Ranjan Kumar Behera, Kshira Sagar Sahoo, Bibhudatta Sahoo, Amir H. Gandomi. Emergence of serverless computing in cloud: A state-of-the-art review of challenges and opportunities. Machine Learning with Applications. V. 24, 2026, 100862. DOI: https://doi.org/10.1016/j.mlwa.2026.100862.
3. Chinmaya Kumar Dehury, Satish Narayana Srirama, Chapter 9 - Continuous testing. Mastering DevOps. Morgan Kaufmann. 2026, pp. 193-215. DOI: https://doi.org/10.1016/B978-0-443-45032-7.00009-2.

4. Корнеев Н.В., Лазорин Д.С. Паттерн для обеспечения безопасности веб-приложения при угрозе XSS атак в облачной инфраструктуре. Вопросы кибербезопасности. 2024, № 6(64), с. 76-84. DOI: https://doi.org/10.21681/2311-3456-2024-6-76-84.
Korneev N.V., Lazorin D.S. Pattern for securing web applications against XSS attacks in cloud infrastructure. Cybersecurity issues. 2024, no. 6(64), pp. 76-84. DOI: https://doi.org/10.21681/2311-3456-2024-6-76-84 (in Russian).
5. Корнеев Н.В., Котрини Е.С. Паттерн для обеспечения безопасности приложения при угрозе модификации модели машинного обучения. Вопросы кибербезопасности. 2025, № 1(65), с. 117-127. DOI: https://doi.org/10.21681/2311-3456-2025-1-117-127.
Korneev N.V., Kotrini E.S. Pattern for securing applications under threat of modification machine learning model. Cybersecurity issues. 2025, v. 1(65), pp. 117-127. DOI: https://doi.org/10.21681/2311-3456-2025-1-117-127 (in Russian).
6. Arpita Patel, James Halgren, Zach Wills et al. NextGen In A Box (NGIAB): Open-Source containerization of the NextGen framework to enable community-driven hydrology modeling. Environmental Modelling & Software. V. 193, 2025, 106666. DOI: https://doi.org/10.1016/j.envsoft.2025.106666.
7. Faezeh Amou Najafabadi, Justus Bogner, Ilias Gerostathopoulos, Patricia Lago. An architectural perspective on MLOps: Structures, processes, tools, and stakeholders. Information and Software Technology. V. 193, 2026, 108029. DOI: https://doi.org/10.1016/j.infsof.2026.108029.
8. Akshit Raj Patel, Sulabh Tyagi, Deepak Sharma. DevOps research across a decade: Patterns, trends, and insights with LDA. Information and Software Technology. V. 195, 2026, 108118. DOI: https://doi.org/10.1016/j.infsof.2026.108118.
9. Xinrui Zhang, Pincan Zhao, Jason Jaskolka. Navigating the DevOps landscape. Journal of Systems and Software. V. 223, 2025, 112331. DOI: https://doi.org/10.1016/j.jss.2024.112331.
10. Rubén Grande, Aurora Vizcaíno, Félix O. García. Is it worth adopting DevOps practices in Global Software Engineering? Possible challenges and benefits. Computer Standards & Interfaces. V. 87, 2024, 103767. DOI: https://doi.org/10.1016/j.csi.2023.103767.
11. Блинов А.В., Беззатеев С.В. DevSecOps: объединение процессов разработки и безопасности. Вопросы кибербезопасности. 2025, № 2(66), с. 78-89. DOI: https://doi.org/10.21681/2311-3456-2025-2-78-89.
Blinov A.V., Bezzateev S.V. DevSecOps: unifying development and security processes. Cybersecurity issues. 2025, V. 2(66), pp. 78-89. DOI: https://doi.org/10.21681/2311-3456-2025-2-78-89 (in Russian).
12. Adi Saputra, Erma Suryani, Nur Aini Rakhmawati. Optimizing AI development efficiency through MLOps and security integration: A measurement approach using data envelopment analysis (DEA). Future Generation Computer Systems. V. 182, 2026, 108503. DOI: https://doi.org/10.1016/j.future.2026.108503.
13. Maysa Sinan, Mojtaba Shahin, Iqbal Gondal. Implementing and integrating security controls: A practitioners' perspective. Computers & Security. V. 156, 2025, 104516. DOI: https://doi.org/10.1016/j.cose.2025.104516.
14. Saeed, H., Shafi, I., Ahmad, J., Khan, A.A., Khurshaid, T. et al. (2025). Review of Techniques for Integrating Security in Software Development Lifecycle. Computers, Materials & Continua, 82(1), 139-172. DOI: https://doi.org/10.32604/cmc.2024.057587.
15. Betul Gokkaya, Leonardo Aniello, Basel Halak. Software supply chain: A taxonomy of attacks, mitigations and risk assessment strategies. Journal of Information Security and Applications. V. 97, 2026, 104324. DOI: https://doi.org/10.1016/j.jisa.2025.104324.
16. Ann Yi Wong, Eyasu Getahun Chekole, Martín Ochoa, Jianying Zhou. On the Security of Containers: Threat Modeling, Attack Analysis, and Mitigation Strategies. Computers & Security. V. 128, 2023, 103140. DOI: https://doi.org/10.1016/j.cose.2023.103140.
17. Девицына С.Н., Пилькевич П.В. Обеспечение совместимости технических компонентов при создании системы мониторинга инцидентов информационной безопасности. Вопросы кибербезопасности. 2024, № 4(62), с. 38-44. DOI: https://doi.org/10.21681/2311-3456-2024-4-38-44.
Devicyna S.N., Pil`kevich P.V. Method for ensuring compatibility of technical components when creating a system for monitoring information security incidents. Cybersecurity issues. 2024, v. 4(62), pp. 38-44. DOI: https://doi.org/10.21681/2311-3456-2024-4-38-44 (in Russian).
18. John Violos, Fotios Voutsas, Christos Diou, Aris Leivadeas. Detecting application transitions and identifying application types for intent-based network assurance: A machine learning perspective. Computer Networks. V. 274, 2026, 111872. DOI: https://doi.org/10.1016/j.comnet.2025.111872.

Конфликт интересов. *Авторы заявляют об отсутствии конфликта интересов.*
Conflict of interest. *The authors declare no conflict of interest.*

Вклад авторов

Корнеев Н.В.: *концептуализация, разработка методологии исследования, работа с программным обеспечением, написание-рецензирование и редактирование рукописи.*

Шиллер Д.И.: *курирование данных, написание-подготовка черновика рукописи, визуализация, проведение исследования.*

Author Contributions
Korneev N.V.: *conceptualization, methodology, software, writing – review & editing.*
Shiller D.I.: *data curation, writing – original draft, visualization, investigation.*

ИНФОРМАЦИЯ ОБ АВТОРАХ:

Николай Владимирович Корнеев, *д.т.н., доцент; профессор, РГУ нефти и газа (НИУ) им. И.М. Губкина*
e-mail: niccyper@mail.ru,
https://orcid.org/0000-0002-0254-1121,
SPIN-код: 4738-7526,
Researcher ID: L-4822-2016,
Scopus Author ID: 57194347465

Дмитрий Игоревич Шиллер, *магистр, Финансовый университет при Правительстве Российской Федерации; инженер сопровождения, ООО «Кросстех Солюшнс Групп»*
e-mail: dshillier@mail.ru,
https://orcid.org/0009-0004-6773-1200

INFORMATION ABOUT THE AUTHORS:

Nikolay Vladimirovich Korneev, *D.Sc. (Eng.), Associate Professor; Professor, Gubkin Russian State University of Oil and Gas (National Research University)*
e-mail: niccyper@mail.ru,
https://orcid.org/0000-0002-0254-1121,
SPIN-code: 4738-7526,
Researcher ID: L-4822-2016,
Scopus Author ID: 57194347465

Dmitry Igorevich Shiller, *Master, Financial University under the Government of the Russian Federation; Software Maintenance Engineer, Crosstech Solutions Group LLS*
e-mail: dshillier@mail.ru,
https://orcid.org/0009-0004-6773-1200

Статья поступила в редакцию 12.06.2026; одобрена после рецензирования 12.07.2026; принята к публикации 27.07.2026
The article was submitted 12.06.2026; approved after reviewing 12.07.2026; accepted for publication 27.07.2026