DevSecOps: ОБЪЕДИНЕНИЕ ПРОЦЕССОВ РАЗРАБОТКИ И БЕЗОПАСНОСТИ

Блинов А. В.[1], Беззатеев С. В.[2]

DOI: 10.21681/2311-3456-2025-2-78-89

Цель исследования: *целью исследования является изучение и описание концепции DevSecOps, её структуры и ключевых компонентов, а также разработка упрощенной модели зрелости DevSecOps. Данная модель может быть использована организациями для оценки текущего уровня зрелости DevSecOps и определения приоритетных направлений для поэтапного внедрения практик безопасной разработки программного обеспечения.*

Методы: *применяется аналитический подход к интеграции безопасности в процессы DevOps с использованием международных стандартов и практик при разработке модели зрелости DevSecOps.*

Полученные результаты: *выявлено, что DevSecOps позволяет объединить процессы разработки, эксплуатации и безопасности, обеспечивая снижение киберрисков за счет интеграции мероприятий по защите на ранних стадиях жизненного цикла программного обеспечения. Описаны три ключевых домена DevSecOps: технологии, процессы и люди, которые формируют основу для перехода к безопасной разработке. Предложенная модель зрелости включает три уровня и 24 активности, которые могут быть использованы для самооценки организаций и создания стратегий внедрения. Также представлены метрики, позволяющие отслеживать прогресс и оценивать эффективность DevSecOps-практик, включая время обнаружения и устранения уязвимостей, а также коэффициенты раннего выявления и производительности.*

Практическая значимость: *разработана упрощённая модель зрелости DevSecOps, обеспечивающая структурированный подход к внедрению безопасной разработки. Впервые предложены комплексные метрики для мониторинга DevSecOps, позволяющие организациям системно подходить к управлению безопасностью и минимизации рисков.*

Ключевые слова: *информационная безопасность, DevSecOps, безопасная разработка ПО, непрерывное тестирование, интеграция безопасности, автоматизация процессов безопасности.*

Введение

Современные организации сталкиваются с необходимостью быстрого выпуска программного обеспечения, что требует постоянного повышения эффективности процессов разработки и обеспечения безопасности. Модель DevOps способствует ускорению разработки и развертывания, но недостаточно учитывает вопросы безопасности, что создает риск появления уязвимостей и кибератак. В ответ на эти вызовы возникла концепция DevSecOps, которая интегрирует безопасность на всех этапах жизненного цикла программного обеспечения. DevSecOps позволяет не только поддерживать высокую скорость разработки, но и минимизировать потенциальные угрозы, делая безопасность общей ответственностью всех участников процесса. Актуальность данной темы обусловлена ростом числа кибератак и необходимостью создания гибких и безопасных процессов разработки, что особенно важно в условиях цифровой трансформации бизнеса.

1. Введение в DevSecOps

1.1. Определение DevSecOps

DevSecOps — это сокращение от «разработка, безопасность и операции» [1]. Это модель разработки программного обеспечения, в которой эти три команды работают в тесном сотрудничестве и в синхронизированном режиме. Можно сказать, что команда DevSecOps — это гибкая, кросс-функциональная команда DevOps [2], которая интегрирует практики безопасности в свои процессы для создания безопасных программных продуктов и цифровых услуг. Другими словами, DevSecOps — это DevOps, реализованный с учётом безопасности [3].

Цель DevSecOps[3] заключается в том, чтобы сделать безопасность общей ответственностью всех участников процесса, сохраняя при этом ту же скорость и масштаб работы, что и в DevOps-процессах непрерывной интеграции и развертывания (CI/CD). Добавление безопасности приложений в DevOps представляет собой серьёзную задачу, поскольку практики безопасности часто становятся «узким местом» на конвейере разработки программного обеспечения. Однако, по мере роста киберугроз, безопасные процессы разработки программного обеспечения становятся всё более важными [4].

Решением является переосмысление и перестройка интеграции практик безопасности в DevOps (рис. 1).

1 Блинов Алексей Валентинович, аспирант 3-го года обучения факультета безопасности информационных технологий Университета ИТМО, Санкт-Петербург, Россия. Orcid.org/0009-0007-2607-2087. E-mail: a.blin0v@mail.ru

2 Беззатеев Сергей Валентинович, доктор технических наук, заведующий кафедрой Информационной безопасности Государственного университета аэрокосмического приборостроения, профессор факультета безопасности информационных технологий Университета ИТМО, Санкт-Петербург, Россия. Orcid.org/0000-0002-0924-6221. E-mail: bsv@guap.ru

3 DevSecOps Manifesto – https://www.devsecops.org/ (дата обращения 28.11.2024)

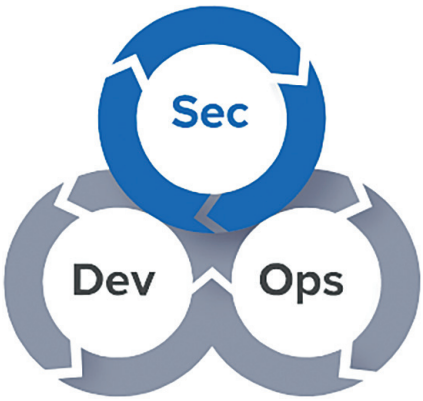

Рис. 1. DevSecOps

Безопасность должна быть неотъемлемой частью DevOps благодаря таким практикам, как безопасное проектирование, ревизия кода, автоматизированное тестирование и тестирование на проникновение. Однако, в реальной жизни такие изолированные практики безопасности DevOps не могут полностью обеспечить защиту, хотя и могут внести свой вклад. Команды DevOps должны принять образ мышления, ориентированный на безопасность, и применять принципы безопасности в своих повседневных гибких действиях.

Сегодня программное обеспечение выпускается в коротких спринтах, поэтому становится крайне важно проводить тестирование безопасности приложений на протяжении всего цикла разработки, не замедляя при этом процессы инженерной работы [5]. В идеале мероприятия по безопасности должны начинаться как можно раньше в жизненном цикле программного обеспечения — это называется «смещением влево» безопасности.

Модель DevSecOps объединяет DevOps и непрерывное тестирование безопасности [6]. Рис. 2

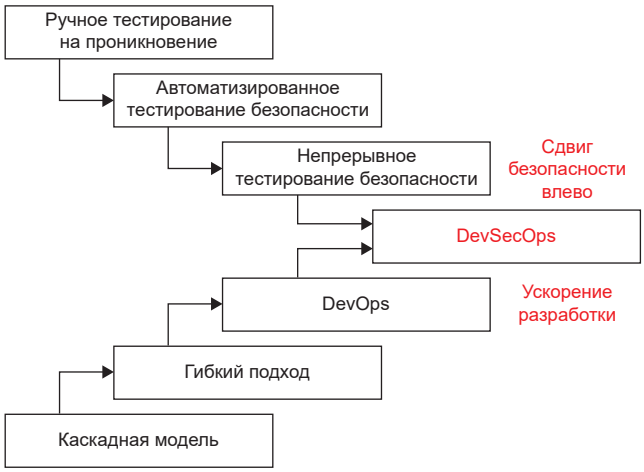

Рис. 2. Переход к DevSecOps

иллюстрирует эволюцию процессов разработки программного обеспечения и безопасности, ведущую к модели DevSecOps. Она показывает, как методы безопасности и разработки постепенно переходят к более интегрированному и ускоренному подходу.

1. Ручное тестирование на проникновение эволюционирует в автоматизированное тестирование безопасности и затем переходит в непрерывное тестирование безопасности. Это указывает на сдвиг безопасности влево, что подразумевает интеграцию мероприятий по безопасности на более ранних этапах жизненного цикла разработки.
2. Одновременно с этим, традиционная методология разработки каскадная модель (Waterfall[4]) трансформируется в гибкий подход (Agile[5]) [7], который в свою очередь приводит к DevOps. Это демонстрирует ускорение разработки, поскольку DevOps способствует более быстрой доставке и интеграции кода.
3. DevSecOps объединяет эти процессы, сочетая DevOps и непрерывное тестирование безопасности, чтобы создать единый, согласованный и безопасный подход к разработке программного обеспечения.

DevSecOps предоставляет разработчикам возможность учитывать безопасность на этапе проектирования системы и при написании кода, а также возможность тестировать программное обеспечение на наличие проблем с безопасностью перед его развертыванием. Кроме того, DevSecOps гарантирует, что у команд разработки есть план оперативного решения вопросов безопасности в случае, если они возникнут после развертывания.

1.2. Структура DevSecOps

DevSecOps можно определить как сочетание технологий, процессов и людей. Именно люди составляют команды Dev (разработки), Ops (операции) и Sec (безопасности). Эти команды следуют принципам Agile в своей повседневной работе. Специалисты из всех трёх команд используют технологические инструменты и работают в соответствии с установленными процессами.

1.2.1. Технология

Технология DevSecOps в целом охватывает безопасность приложений, данных и инфраструктуры.

Безопасность приложений (AppSec) направлена на повышение безопасности программного обеспечения за счёт применения принципов безопасного проектирования к создаваемым приложениям, тестирования, поиска и устранения уязвимостей

4 Waterfall Model – Software Engineering – https://www.geeksforgeeks.org/waterfall-model/ (дата обращения 28.11.2024).

5 What is Agile Methodology? – https://www.geeksforgeeks.org/what-is-agile-methodology/ (дата обращения 28.11.2024).

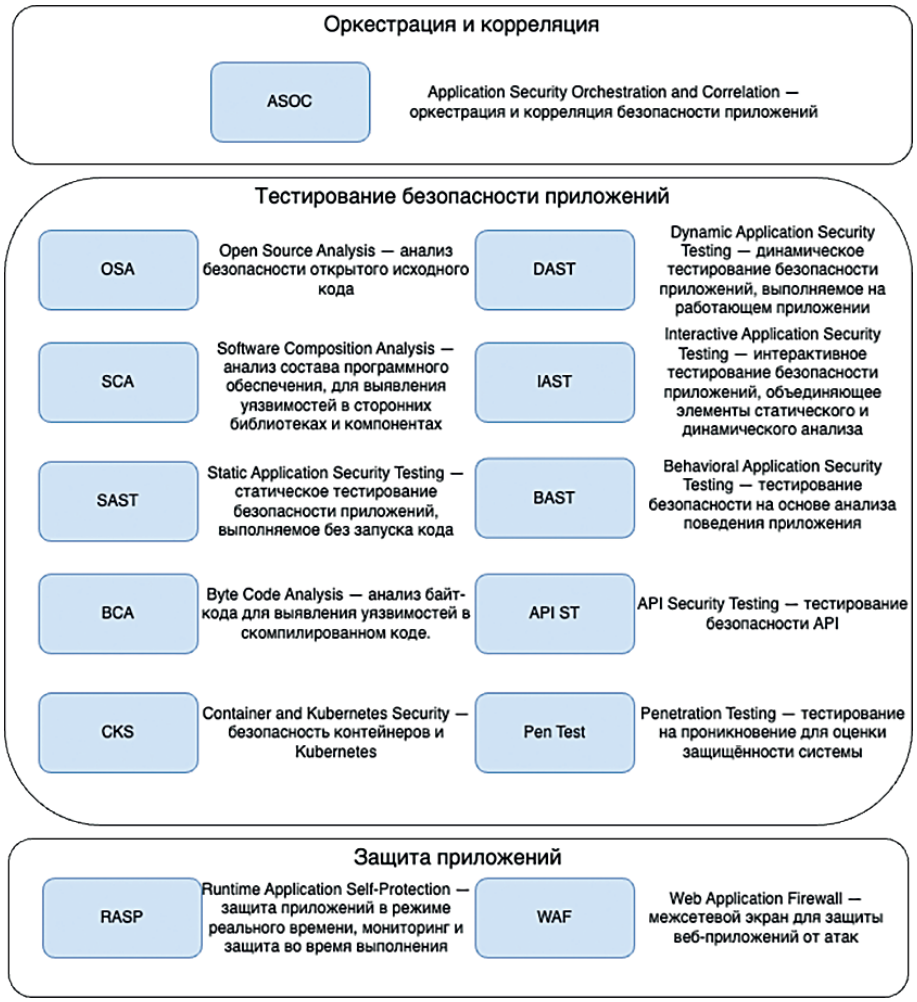

Рис. 3. Практики безопасности приложений

в коде, а также защиты приложений на этапе их развертывания.

AppSec помогает командам разработки и безопасности работать более эффективно и масштабировать DevSecOps с помощью автоматизации, интеграции и сотрудничества, чтобы они могли продолжать создавать инновационные приложения. DevSecOps предполагает программный подход, ориентированный на непрерывную интеграцию AppSec в DevOps.

На сегодняшний день цепочка инструментов безопасности приложений в основном сосредоточена на тестировании. Именно поэтому используется термин тестирование безопасности приложений (Application Security Testing (AST)) [8], охватывающий большинство технологических областей безопасности, перечисленных ниже. Каждая область безопасности приложений называется практикой безопасности приложений (Рис. 3). Все практики делятся на три группы: AST, Оркестрация и корреляция и Защита приложений.

ASOC выделяется среди всех элементов на рисунке, так как его основная цель — координировать все виды тестирования безопасности и коррелировать собранные данные, группируя выявленные уязвимости вместе. ASOC помогает обеспечить целостный подход к управлению безопасностью, оптимизируя процессы и улучшая восприятие угроз.

RASP и WAF обеспечивают дополнительные уровни защиты, когда приложение находится в рабочей (продукционной) среде. Их основная задача — защищать приложения от атак в режиме реального времени, в то время как AST сосредотачивается на выявлении и устранении уязвимостей в приложении до его развертывания. Это позволяет обеспечить многоуровневую защиту, начиная от процесса разработки и заканчивая реальной эксплуатацией приложения.

1.2.2. Процесс

Необходимо внедрить интегрированный процесс, в рамках которого команда инженеров по безопасности сможет легко выполнять тесты безопасности приложений, выявлять дефекты безопасности, передавать их команде разработчиков, проверять корректность исправлений и закрывать устранённые дефекты [9].

Рис. 4 иллюстрирует интегрированный процесс взаимодействия между разработкой приложений и безопасностью приложений.

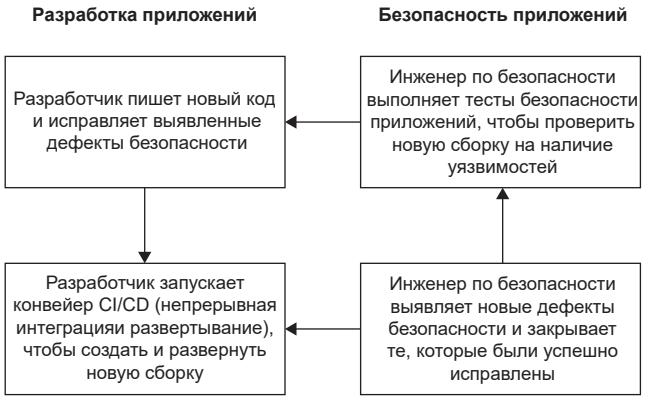

Рис. 4. Процесс взаимодействия

Процесс безопасного жизненного цикла разработки программного обеспечения (SSDL[6]) предполагает внесение необходимых изменений и ограничений в существующий процесс инженерии программного обеспечения [10]. Процедуры, регламенты, требования и стандарты безопасной разработки ПО должны быть задокументированы, опубликованы для всех ключевых заинтересованных сторон и отражены в политиках компании.

Следует применять организационные и технические меры для обеспечения безопасного процесса разработки ПО. Внедрение лучших практик безопасной разработки ведёт к непрерывному улучшению процесса. Необходимо определить, задокументировать и измерить количественные метрики безопасности и ключевые показатели эффективности (KPI).

1.2.3. Люди

Создание группы по безопасности программного обеспечения (Software Security Group (SSG)) является одним из ключевых компонентов организационной готовности к DevSecOps. SSG накапливает и распространяет знания по всей организации, а также определяет новые роли и обязанности для руководства и необходимых технических специалистов.

Индустрия разработки программного обеспечения сталкивается с нехваткой качественных специалистов по безопасности, чтобы поддерживать растущие масштабы разработки [11]. Поэтому критически важно найти сотрудников, которые готовы сосредоточить свою карьеру на вопросах безопасности. Они будут помогать другим членам команды улучшать безопасность разрабатываемых программных продуктов и услуг.

6 Microsoft Security Development Lifecycle (SDL) – https://www.microsoft.com/en-us/securityengineering/sdl/practices (дата обращения 28.11.2024).

Члены команд разработки, команды безопасности и другие заинтересованные стороны, участвующие в SSDL, должны пройти обучение по процедурам, регламентам, процессам, требованиям и стандартам безопасной разработки программного обеспечения.

1.3. Культура безопасной разработки программного обеспечения

Существуют инструменты и процессы, которые могут помочь вам перейти к организации в стиле DevSecOps [12]. Однако в конечном итоге DevSecOps — это культура, соответствующая четырём принципам, изложенным в Манифесте безопасной разработки в Agile[7]:

1. Опора на разработчиков и тестировщиков больше, чем на специалистов по безопасности. Основная ответственность за безопасность должна лежать на команде разработки, работающей над программным обеспечением, а не исключительно на специалистах по безопасности.
2. Интеграция безопасности в процесс работы, а не после его завершения. Безопасность должна быть встроена в существующий процесс повседневной разработки и тестирования, который выполняется командой разработки.
3. Реализация функций безопасности, а не добавление отдельных функций. Предпочтительнее вовлекать опытных специалистов для реализации специфических аспектов безопасности внутри программного обеспечения, не предоставляя пользователям контроль над функциями безопасности.
4. Управление рисками, а не только исправление ошибок. Исправление ошибок безопасности само по себе не является правильным подходом. Необходимо сосредоточиться на управлении рисками и минимизации рисков для бизнеса и пользователей.

Компании, ориентированные на инженерные разработки, меняют свою рабочую культуру, внедряя свои знания и опыт в области безопасности как часть кода, что приносит пользу всем.

2. Упрощенная модель зрелости DevSecOps

Представим упрощённую модель зрелости DevSecOps для создания дорожной карты внедрения DevSecOps в масштабах всей организации. Это хороший первый шаг перед более детализированным анализом на основе фреймворка BSIMM (Built Security In Maturity Model[8]) — более сложной модели зрелости.

7 Agile Manifest – https://agilemanifesto.org/iso/ru/manifesto.html (дата обращения 28.11.2024).

8 What is BSIMM? – https://www.blackduck.com/glossary/what-is-bsimm.html (дата обращения 28.11.2024).

Модель зрелости DevSecOps, описанная ниже, может быть использована организациями для оценки текущего уровня зрелости и приоритизации различных элементов DevSecOps для поэтапного внедрения.

Инициатива по безопасности программного обеспечения должна быть запущена для внедрения и постоянного обновления практик безопасной разработки ПО, накопления знаний и создания группы по безопасности ПО.

Основные принципы Инициативы по безопасности программного обеспечения:

1. Цели: внедрение и поддержка безопасной разработки ПО должны быть направлены на предотвращение существующих угроз и известных проблем, а также на минимизацию риска появления новых. Стоимость Инициативы по безопасности ПО должна быть разумной и пропорциональной возможным потерям от корпоративных рисков и угроз безопасности.
2. Процесс: процедуры, регламенты, процессы, требования и стандарты безопасной разработки ПО должны быть задокументированы, опубликованы для всех ключевых заинтересованных сторон и отражены в политике компании. Организационные и технические меры для безопасного процесса разработки должны применяться. Внедрение лучших практик безопасной разработки ведёт к постоянному улучшению процессов.
3. Организация: внедрение SSDL должно поддерживаться всеми ключевыми заинтересованными сторонами (топ-менеджментом, командами разработки и командами безопасности). Сотрудники несут ответственность за внедрение и использование практик безопасной разработки в соответствии с их должностными обязанностями. Члены команд разработки, команд безопасности и другие заинтересованные стороны, вовлеченные в SSDL, должны пройти обучение по процессу безопасной разработки ПО.

Для создания целостного подхода к безопасности в жизненном цикле разработки ПО необходимо построить чёткий путь к целевому состоянию Инициативы по безопасности ПО, собрать данные и сформировать детализированную стратегию.

Начальным этапом является оценка текущего состояния практик безопасности приложений. Модель зрелости DevSecOps поможет провести самооценку текущих мероприятий по безопасности, определить уровень зрелости для каждой области и обозначить возможное целевое состояние практик безопасности приложений.

Таблица 1.

Поведенческие индикаторы

	Начальный уровень зрелости	Средний уровень зрелости	Полный уровень зрелости
Технологии			
Оркестрация и корреляция			
ASOC	○	◑	●
Тестирование безопасности приложений			
SAST	◑	●	●
OSA	◑	●	●
SCA	○	◑	●
DAST	◑	●	●
API ST	○	◑	●
BCA	○	◑	●
CKS	○	◑	●
IAST	○	○	●
BAST	○	◑	●
Pen Test	●	●	●
Защита приложений			
RASP	○	◑	●
WAF	◑	◑	●
Процессы			
Дорожная карта DevSecOps	●	●	●
Процесс SSDL	◑	●	●
Отчёты	◑	◑	●
Метрики	○	◑	●
Платформа DevSecOps	○	◑	●
Многоразовые компоненты безопасности	◑	◑	●
Люди			
Поддержка на высшем уровне	◑	●	●
Бизнес-партнер по безопасности приложений	◑	●	●
Чемпионы безопасности	◑	●	●
Экспертиза по безопасности ПО	◑	◑	●
Осведомленность и обучение	◑	◑	●
Группа по безопасности ПО	○	◑	●

DevSecOps можно определить как сочетание технологий, процессов и обучения людей. Модель оценки зрелости DevSecOps состоит из трёх доменов:

- Технологии
- Процессы
- Люди

Она включает 24 активности и разделена на три уровня зрелости. Для каждого домена и каждого уровня в таблице 1 описаны типичные поведенческие индикаторы.

В модели зрелости DevSecOps степень выполнения каждой активности для каждого уровня зрелости отображается графически в виде круга. Белый круг (○) означает, что работа в этом направлении ещё не началась. Наполовину заполненный круг (◑) показывает, насколько продвинуто внедрение данной активности. Полностью заполненный круг (●) соответствует полной реализации активности. Таким образом, модель является не только качественной, но и количественной, позволяя оценивать прогресс.

Теперь кратко опишем каждую из активностей модели.

2.1. Технологии

Этот домен охватывает все практики технологий безопасности приложений, разделённые на три группы: Оркестрация и корреляция, Тестирование безопасности приложений и Защита приложений (Рис. 5)

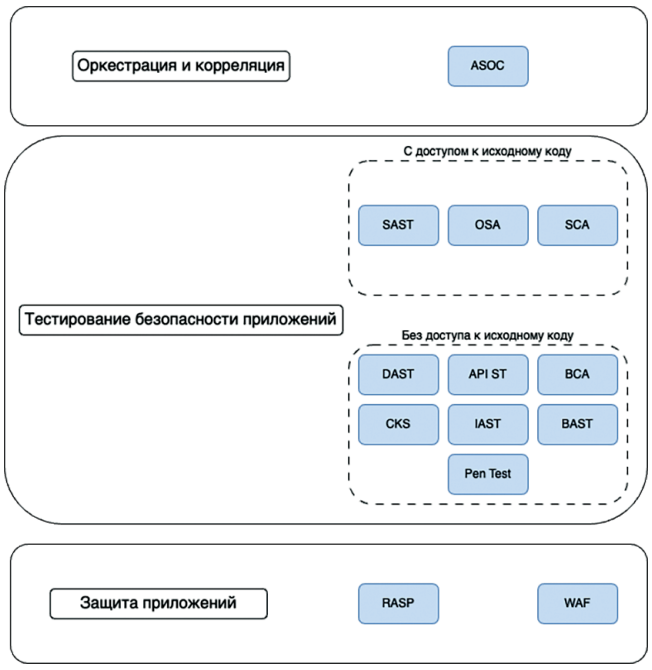

Рис. 5. Технологии

2.2. Процессы

Активности в этом домене включают:

- Дорожная карта – внедрение процессов и инструментов безопасности приложений в организацию по разработке ПО — это сложная трансформационная программа. Такая программа должна включать дорожную карту, которая определяет приложения, людей, процессы и технологии, планируемые для внедрения на каждом этапе. Дорожная карта помогает управлять последовательностью действий и приоритетами в процессе перехода к DevSecOps.
- Процесс SSDL – целевой процесс SSDL определяется с учётом необходимых изменений и ограничений существующего процесса разработки ПО. Процедуры, регламенты, требования и стандарты безопасной разработки ПО должны быть задокументированы и опубликованы для всех ключевых заинтересованных сторон, а также отражены в политиках компании. Следует четко определить и задокументировать роли и обязанности бизнес-заказчиков, команд разработки, поставщиков ПО и группы безопасности ПО.
- Отчеты – регулярное составление отчетов о прогрессе, достижениях, рисках и планах по безопасности приложений и DevSecOps активности должно быть организовано на разных уровнях: на уровне проекта и отдела отчеты предоставляют чемпионы безопасности, на уровне всей организации — SSG. Это способствует поддержанию прозрачности и видимости процесса для всех вовлечённых сторон.
- Метрики – сбор данных DevSecOps и визуализация метрик позволяют управлять полным процессом безопасной разработки ПО, обеспечивая прозрачность. Это также создаёт необходимую основу для управления процессом, основанного на данных, с использованием технологий машинного обучения в будущем. Необходимо определить и задокументировать количественные метрики безопасности и ключевые показатели эффективности.
- Платформа DevSecOps – создание платформы DevSecOps предполагает полную автоматизацию процесса разработки программного обеспечения на основе процессов DevSecOps. Это включает автоматизацию тестирования, развертывания, контроля безопасности и других ключевых этапов жизненного цикла разработки.
- Многоразовые компоненты безопасности – создание библиотеки компонентов программного обеспечения, проверенных на безопасность, для повторного использования внутри организации, а также рекомендации по их применению значительно облегчат процесс разработки приложений. Повторное использование проверенных компонентов повышает безопасность и ускоряет процесс разработки.

2.3. Люди

Данный домен включает следующие активности:

- Поддержка на высшем уровне – внедрение SSDL должно поддерживаться всеми заинтересованными сторонами, начиная с высшего руководства. Поддержка со стороны старших руководителей необходима для определения и утверждения требуемого бюджета, управления операциями, привлечения ресурсов и предоставления политического прикрытия для инициативы по безопасности ПО. Назначение старшего руководителя, ответственного за безопасность программного обеспечения, помогает решать вопросы управления, связанные с ответственностью и наделением полномочиями.
- Бизнес-партнер по безопасности приложений – это технический, идеологический и методологический эксперт в области безопасности приложений, который взаимодействует со всеми заинтересованными сторонами, включая руководство, клиентов, SSG и команды разработки. Бизнес-партнер по безопасности приложений является ключевым двигателем всей Инициативы по безопасности приложений и должен охватывать вопросы стратегии, процессов и DevSecOps. Этот человек должен быть увлечён темой безопасности приложений, обладать хорошими лидерскими качествами и уметь эффективно презентовать свои идеи.
- Чемпионы безопасности – это проводники культуры безопасной разработки ПО, ответственные за безопасность приложений в командах разработки. Они не являются частью SSG, но формируют сообщество, которое поддерживает безопасность и обеспечивает связь с основными группами.
- Экспертиза по безопасности ПО – экспертиза в области безопасности программного обеспечения означает накопление знаний сотрудниками компании через обучение процессу безопасной разработки ПО, а также повышение технологической зрелости сотрудников в разработке DevSecOps. Повышение экспертизы в области безопасности ПО описывается как усложнение используемых технологий AppSec и DevSecOps, расширение задач, решаемых с их помощью, и увеличение процента сотрудников, квалифицированных в области AppSec и DevSecOps.
- Осведомленность и обучение – члены команд разработки ПО, команд безопасности и другие заинтересованные стороны, вовлечённые в SSDL, должны быть обучены процессу безопасной разработки ПО. Непрерывная практика обучения является важной для организаций, занимающихся разработкой ПО, стремящихся достичь соответствующего уровня зрелости и минимизировать внедрение уязвимостей в свои приложения и сервисы. Она включает программы повышения осведомлённости и технические тренинги.
  - **Первый шаг в реализации программы повышения осведомлённости** – это определение процесса для передачи общих требований по безопасности, практик и концепций безопасности приложений командам разработки. Следующим шагом является обеспечение широкого охвата и осведомлённости среди разработчиков ПО.
  - **Обучающие курсы** разработаны, чтобы помочь командам разработки создавать более безопасный код. Курсы, проводимые практикующими экспертами с многолетним опытом разработки на различных платформах, предоставляют рабочие знания об угрозах и соответствующих мерах противодействия, сочетая концепции безопасности и практическое обучение разработке.
- Группа по безопасности ПО – создание специализированной SSG является одним из ключевых компонентов организационной трансформации. SSG накапливает и распространяет знания по всей организации и определяет новые роли и обязанности для руководства и необходимых технических специалистов.

Модель зрелости DevSecOps может служить дорожной картой для внедрения Инициативы по безопасности ПО в организациях. Для достижения следующего уровня зрелости необходимо развивать соответствующие активности. Это может показаться сложной задачей, но безопасность приложений и, в конечном итоге, безопасность клиентов оправдывают все усилия.

3. Метрики DevSecOps

Метрики являются критически важной частью DevSecOps. Метрики DevSecOps предоставляют всестороннее представление о текущем состоянии безопасности приложений и служат важными индикаторами для непрерывного улучшения уровня безопасности ПО в организации с течением времени. Обладая этими метриками и активно используя их для анализа и совершенствования реализации программы DevSecOps, организации могут более эффективно защищать данные клиентов и снижать операционные риски.

Метрики помогают:

- Оценивать текущую ситуацию в области безопасности приложений.
- Отслеживать прогресс и измерять эффективность внедрения процессов и инструментов DevSecOps.

- Принимать решения на основе данных, что позволяет своевременно выявлять и устранять слабые места в процессе разработки и безопасности.

3.1. Определения

Уязвимость – это ошибка безопасности, изъян или слабое место в программном обеспечении, которое может быть использовано для нарушения функциональности или предоставления несанкционированного доступа к ресурсам приложения, делая его уязвимым.

Критичность – это свойство конкретной уязвимости, определяющее, насколько она важна с технической точки зрения.

Определение критичности (Severity) основывается на двух аспектах:

1. Воздействие эксплуатации уязвимости, которое оценивается по шкале: Высокое/Среднее/Низкое (High/ Medium/Low).
2. Вероятность эксплуатации уязвимости, также оцениваемая по шкале: Высокая / Средняя / Низкая (High/ Medium/Low).

Оценка воздействия (Impact) включает следующие факторы:

- Влияние на конфиденциальность — оценивается на основе типа и объема цифровых активов, которые могут быть скомпрометированы.
- Влияние на целостность — оценивается по объему функциональности, которая ведёт себя не так, как должна.
- Влияние на доступность — зависит от общей доступности приложения/цифрового сервиса и их конкретных бизнес-функций.

Оценка вероятности эксплуатации уязвимости основывается на следующих факторах:

- Распространённость этого типа уязвимостей (например, наличие в списках OWASP Top-10, OWASP Mobile Top-10, SANS25 и др.).
- Общая сложность эксплуатации уязвимости.
- Скорость эксплуатации уязвимости.

Возможные значения критичности уязвимости:

- Критическая (Critical)
- Высокая (High)
- Средняя (Medium)
- Низкая (Low)

На рис. 6 представлена матрицу оценки рисков, которая показывает взаимосвязь вероятности (Probability) эксплуатации уязвимости и её воздействия (Impact).

Приоритет — это свойство конкретной уязвимости, которое определяет её критичность с точки зрения бизнеса и задаёт её приоритет в бэклоге дефектов безопасности для исправления командой разработки. Возможные значения: Блокер/Критичная/Значительная/Незначительная/Информационная (Blocker/Critical/Major/Minor/Info).

Обнаруженные уязвимости — это уязвимости, выявленные (обнаруженные) в ходе сканирования безопасности в рамках непрерывного цикла DevSecOps. Уязвимости, обнаруженные инструментами безопасности приложений в ходе сканирования, также называются «проблемами безопасности».

Открытые уязвимости — это уязвимости, которые были обнаружены и зарегистрированы в системе отслеживания дефектов. Уязвимости, зарегистрированные в системе отслеживания дефектов, также называются «дефектами безопасности». Один дефект безопасности может включать одну или несколько проблем безопасности. Несколько обнаруженных уязвимостей (проблем безопасности) могут быть технически объединены в один дефект безопасности в зависимости от их природы и происхождения.

Разрешённые уязвимости — это уязвимости, которые команда разработчиков отметила, как исправленные в системе отслеживания дефектов, но которые ещё не были повторно протестированы командами безопасности.

Подтверждённые уязвимости — это уязвимости, которые были исправлены, повторно протестированы в ходе нового сканирования безопасности и подтверждены как устранённые.

Исправленные уязвимости — это подтверждённые уязвимости, которые были удалены из рабочей среды. Это относится к уязвимостям, которые ранее попали в рабочую среду и теперь полностью устранены.

Уязвимости, попавшие в продакшн — это уязвимости, которые были обнаружены и/или зарегистрированы, но не исправлены к моменту релиза, и, следовательно, стали доступны в рабочей среде или в сборке, выпущенной для клиентов.

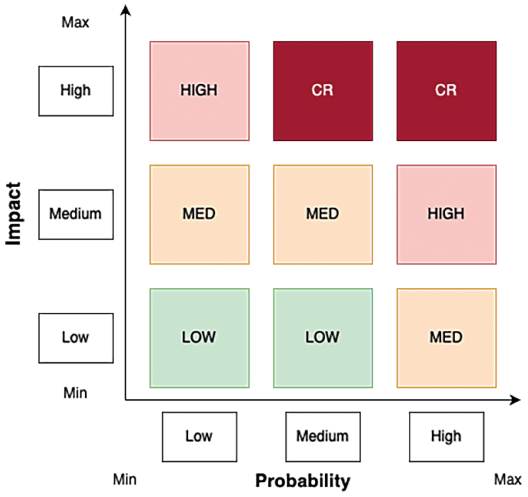

Рис. 6. Матрица оценки риска

Шлюзы безопасности определяют набор критериев безопасности, которым должен соответствовать новый код приложения перед его развертыванием в конкретной инженерной среде (DEV/TEST/STAGE/PROD).

3.2. Скорость DevSecOps

3.2.1. Среднее время в рабочей среде

Среднее время в рабочей среде описывает среднюю продолжительность времени, в течение которого уязвимость находится в рабочей среде до её устранения и исправления (удаления из рабочей среды). Эта метрика применяется к уязвимостям, которые уже были устранены после попадания в рабочую среду. Измеряется в момент, когда код выпускается в рабочую среду.

Среднее время в рабочей среде = Длительность (в днях) с момента, когда уязвимость попала в рабочую среду с определённым релизом, до момента, когда исправленный код был развернут в рабочей среде как патч или с следующим релизом.

Среднее время в рабочей среде [Критичность] = Среднее время в рабочей среде по критичности (Критическая, Высокая, Средняя, Низкая). Обычно эта метрика измеряется для критических и высоких уязвимостей, которые по стандартам компании не должны присутствовать в рабочей среде.

Идеально, если среднее время в рабочей среде равно нулю для критических и высоких уязвимостей. Если эта метрика больше одной длительности релиза, это означает, что (a) такие уязвимости не были обнаружены в нескольких релизах подряд или (b) на их исправление, потребовалось гораздо больше времени, чем длительность одного релиза. Тенденция среднего время в рабочей среде отслеживается со временем для анализа прогресса.

3.2.2 Среднее время обнаружения

Среднее время обнаружения — это время, необходимое для обнаружения уязвимости после того, как новый код с этой уязвимостью был добавлен. Эта метрика применяется к обнаруженным уязвимостям.

Среднее время обнаружения = Длительность (в днях) с момента, когда новый код с уязвимостью был добавлен, до момента, когда эта уязвимость была обнаружена.

Цель — обнаруживать уязвимости как можно раньше, а не ближе к концу релиза. Тенденция среднего время обнаружения отслеживается с течением времени для анализа прогресса. Среднее время обнаружения должно быть на низком уровне, чтобы реализовать парадигму Shift-Left в разработке ПО.

Идеально, если среднее время обнаружения не превышает 30 % длительности релиза. Если среднее время обнаружения больше, чем длительность релиза, это означает, что уязвимости попадают в рабочую среду, и безопасность приложений ухудшается. Когда среднее время обнаружения меньше 30 % длительности релиза, уязвимости выявляются в начале разработки, и есть достаточно времени для их исправления в рамках того же релиза.

3.2.3. Среднее время устранения

Среднее время устранения — это среднее время, необходимое команде разработки для устранения дефекта безопасности. Эта метрика применяется к уязвимостям, которые были устранены.

Среднее время устранения = Длительность (в днях) с момента обнаружения уязвимости до момента её устранения.

Цель — быстро устранять дефекты в соответствии с их приоритетом и не допускать не устранённые дефекты в рабочую среду. Среднее время устранения должно быть меньше 30 % длительности релиза. Когда среднее время устранения меньше 30 % длительности релиза, уязвимости, выявленные в середине релиза, будут устранены в рамках того же релиза перед развертыванием в рабочей среде. Если среднее время устранения больше 30 % длительности релиза, некоторые уязвимости, обнаруженные на поздних этапах релиза, могут не быть исправлены до его завершения и попасть в рабочую среду. Если среднее время устранения больше, чем цикл релиза, все новые уязвимости попадают в рабочую среду, и безопасность приложений становится нестабильной. Тенденция среднего времени устранения отслеживается с течением времени для анализа прогресса.

3.3. Производительность DevSecOps

3.3.1. Коэффициент раннего обнаружения

Коэффициент раннего обнаружения — это процент уязвимостей, обнаруженных на каждом этапе среды, от общего числа уязвимостей, выявленных в релизе.

Коэффициент раннего обнаружения % = Количество уязвимостей, обнаруженных в различных средах (DEV, TEST, STAGE, PROD), делённое на общее количество уязвимостей, выявленных в течение релиза.

Коэффициент раннего обнаружения показывает способность выявлять проблемы безопасности на ранних этапах жизненного цикла разработки. Идеальные значения: Коэффициент раннего обнаружения для DEV – 60 %, TEST – 30 %, STAGE – 10 %, PROD – 0 %.

3.3.2. Коэффициент неудачных конвейеров безопасности

Коэффициент неудачных конвейеров безопасности — это процент незавершённых конвейеров безопасности от общего числа выполненных конвейеров безопасности в релизе.

Коэффициент неудачных конвейеров безопасности = Количество конвейеров безопасности со статусом «неудачный» / Общее количество выполненных конвейеров безопасности за релиз.

Коэффициент неудачных конвейеров безопасности применяется к конвейерам безопасности, инициированным в течение конкретного релиза. Эта метрика указывает на общее качество операций DevSecOps. В идеале она должна быть равна нулю.

3.3.3. Время ожидания сканирования

Время ожидания сканирования – это время ожидания задания на сканирование с момента его добавления в очередь до начала выполнения. Эта метрика применяется ко всем видам сканирования безопасности.

Время ожидания сканирования = Время начала выполнения задания в конвейере минус время добавления этого задания в очередь в конвейере CI/CD безопасности.

Время ожидания сканирования в идеале должно быть равно нулю и не увеличиваться со временем по мере добавления в DevSecOps большего числа приложений.

3.3.4. Время сканирования безопасности

Время сканирования безопасности показывает, сколько времени требуется для выполнения сканирования безопасности.

Время сканирования безопасности = Продолжительность сканирования безопасности с момента запуска до успешного завершения.

Время сканирования безопасности [Практики] = Время сканирования безопасности по практике (например, SAST, DAST, SCA и др.).

Время сканирования безопасности измеряется в часах. Эта метрика может служить индикатором необходимости увеличения вычислительных мощностей для сокращения времени сканирования до разумной продолжительности.

Заключение

В данной статье рассмотрена концепция DevSecOps, представляющая собой подход, объединяющий процессы разработки, безопасности и эксплуатации для создания защищенных программных продуктов. Основной целью DevSecOps является интеграция безопасности на всех этапах жизненного цикла разработки программного обеспечения, что позволяет снизить риски и повысить качество выпускаемого продукта. Приведена структура DevSecOps, включающая три ключевых домена — технологии, процессы и люди, — и предложена модель зрелости, помогающая организациям оценить текущий уровень и выстроить стратегию внедрения DevSecOps.

Показано, что успешная реализация DevSecOps требует тесного взаимодействия команд разработки, безопасности и эксплуатации, использования автоматизированных процессов и непрерывного тестирования. Кроме того, внедрение DevSecOps способствует формированию культуры безопасности, где каждый участник процесса осознает свою роль в защите программного обеспечения.

Для обеспечения эффективного управления безопасностью разработаны метрики DevSecOps, позволяющие оценивать текущую ситуацию, отслеживать прогресс и принимать решения на основе данных. Применение данных инструментов способствует улучшению процессов разработки и повышению общей безопасности приложений.

Таким образом, DevSecOps является важным шагом в развитии современной разработки программного обеспечения, способным повысить устойчивость организаций к киберугрозам и минимизировать риски, связанные с уязвимостями в программных продуктах.

Литература

1. Селиверстов С.Д., Мироненко Ю.В. Обзор методологии DevSecOps и ее ключевых инструментов для внедрения и обеспечения безопасной разработки ПО // Студент года 2024 – сборник статей Международного научно-исследовательского конкурса. Пенза, 2024. С. 107–111.
2. Ганжур М.А., Дьяченко Н.В., Отакулов А.С. Анализ методологий DevOps и DevSecOps // Молодой Исследователь Дона. 2021. № 5 (32). С. 8–10.
3. Kim, G., Humble, J., Debois, P., & Willis, J. (2016). The DevOps Handbook: How to Create World-Class Agility, Reliability & Security in Technology Organizations. IT Revolution Press.
4. Тулеубаева А.А., Норкина А.Н. Современные проблемы информационной безопасности в разработке программного обеспечения // Угрозы и риски финансовой безопасности в контексте цифровой трансформации: Материалы VII Международной научно-практической конференции Международного сетевого института в сфере ПОД/ФТ, Москва, 24 ноября 2021 года. – Москва: Национальный исследовательский ядерный университет «МИФИ», 2021. С. 670–676.
5. Зиновьев, Л.Д., Каледа Р.А. Применение методов DevSecPps для интеграции безопасности в каждый этап жизненного цикла программного обеспечения // Информационные технологии в науке и образовании. Проблемы и перспективы: Сборник статей по материалам XI Всероссийской научно-практической конференции, г. Пенза, 13 марта 2024 года. Пенза: Пензенский государственный университет, 2024. С. 271–273.
6. Reddy Chittibala, D. DevSecOps: Integrating Security into the DevOps Pipeline // International Journal of Science and Research. 2023. № 12(12). С. 2074–2078. DOI 10.21275/sr24304171058.
7. Майорова, Е.В., Соколовская С.А., Черток А.В. Преимущества гибкого подхода для сопровождения проектов разработки программного продукта // Петербургский экономический журнал. 2019. № 4. С. 42–51. DOI 10.25631/PEJ.2019.4.42.51.
8. Фатхи, В.А., Дьяченко Н.В. Тестирование безопасности приложений // Инженерный вестник Дона. 2021. № 5(77). С. 108–120.
9. Кузьмина, С.П. Роль пайплайнов в современной кибербезопасности: автоматизация, защита и реагирование на угрозы // Интернаука. 2024. № 33-1(350). С. 9-10.

10. Горшков А. Г., Пителинский К. В., Михайлов И. А., Медникова З. М., Умерзакова Д. А. Особенности применения концепции SSDLC при разработке и тестировании защищенного программного обеспечения // Информационные технологии в проектировании и производстве. 2024. № 3(195). С. 30–36. DOI: 10.52190/2073-2597_2024_3_30.
11. Naidoo, R. Building Software Applications Securely with DevSecOps: A Socio-Technical Perspective // European Conference on Cyber Warfare and Security. 2022. № 1 (21). С. 198–205. DOI: 10.34190/eccws.21.1.295.
12. Майорова, Е. В., Соколовская С. А., Черток А. В. Обеспечение информационной безопасности при гибком подходе разработки программного продукта // Цифровые технологии обработки и защиты информации: Сборник научных статей / Под редакцией Е. В. Стельмашонок, И. Н. Васильевой. Санкт-Петербург: Санкт-Петербургский государственный экономический университет, 2020. С. 83–92.

DevSecOps: UNIFYING DEVELOPMENT AND SECURITY PROCESSES

Blinov A. V.[9], Bezzateev S. V.[10]

Keywords: *information security, DevSecOps, secure software development, continuous testing, security integration, security process automation, DevOps.*

Research objective: *the objective of this study is to examine and describe the concept of DevSecOps, its structure, and key components, as well as to develop a simplified DevSecOps maturity model. This model can be utilized by organizations to assess their current DevSecOps maturity level and identify priority areas for the phased implementation of secure software development practices.*

Methods: *the research involved analyzing modern approaches to integrating security into DevOps processes, developing a DevSecOps maturity model based on international standards and practices, and creating methodologies for maturity assessment and metrics for monitoring and managing security.*

Results: *the research revealed that DevSecOps unifies development, operations, and security processes, reducing cybersecurity risks by integrating protective measures at the early stages of the software lifecycle. Three key domains of DevSecOps were identified: technology, processes, and people, which form the foundation for transitioning to secure development. The proposed maturity model comprises three levels and 24 activities that organizations can use for self-assessment and strategy development for implementation. Additionally, metrics were introduced to monitor progress and evaluate the effectiveness of DevSecOps practices, including vulnerability detection and remediation time, early detection rates, and performance coefficients.*

Practical significance: *a simplified DevSecOps maturity model was developed, providing a structured approach to implementing secure development practices. For the first time, comprehensive metrics for DevSecOps monitoring were proposed, enabling organizations to adopt a systematic approach to security management and risk minimization.*

References

1. Seliverstov S. D., Mironenko Y. V. Obzor metodologii DevSecOps i ee klyuchevyh instrumentov dlya vnedreniya i obespecheniya bezopasnoj razrabotki PO // Student of the Year 2024 – sbornik statej Mezhdunarodnogo nauchno-issledovatel'skogo konkursa. Penza, 2024. pp. 107–111.
2. Ganzhur M. A., D'yachenko N. V., Otakulov A. S. Analiz metodologij DevOps i DevSecOps // Molodoj Issledovatel' Dona. 2021. № 5 (32). pp. 8–10.
3. Kim, G., Humble, J., Debois, P., & Willis, J. (2016). The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations. IT Revolution Press.
4. Tuleubaeva A. A., Norkina A. N. Sovremennye problemy informacionnoj bezopasnosti v razrabotke programmnogo obespecheniya // Ugrozy i riski finansovoj bezopasnosti v kontekste cifrovoj transformacii: Materialy VII Mezhdunarodnoj nauchno-prakticheskoj konferencii Mezhdunarodnogo setevogo instituta v sfere POD/FT, Moscow, 24.11.2021. – Moskva: Nacional'nyj issledovatel'skij yadernyj universitet «MIFI», 2021. pp. 670–676.
5. Zinov'ev, L. D., Kaleda R. A. Primenenie metodov DevSecPps dlya integracii bezopasnosti v kazhdyj etap zhiznennogo cikla programmnogo obespecheniya // Informacionnye tekhnologii v nauke i obrazovanii. Problemy i perspektivy: Sbornik statej po materialam XI Vserossijskoj nauchno-prakticheskoj konferencii, g. Penza, 13.032.024. Penza: Penzenskij gosudarstvennyj universitet, 2024. pp. 271–273.
6. Reddy Chittibala, D. DevSecOps: Integrating Security into the DevOps Pipeline // International Journal of Science and Research. 2023. № 12(12). pp. 2074–2078. DOI 10.21275/sr24304171058.
7. Majorova, E. V., Sokolovskaya S. A., Chertok A. V. Preimushchestva gibkogo podhoda dlya soprovozhdeniya proektov razrabotki programmnogo produkta // Peterburgskij ekonomicheskij zhurnal. 2019. № 4. pp. 42–51. DOI 10.25631/PEJ.2019.4.42.51.

9 Alexey V. Blinov, Ph.D. student, Faculty of Information Technology Security, ITMO University, St. Petersburg, Russia. Orcid.org/0009-0007-2607-2087. E-mail: a.blin0v@mail.ru

10 Sergey V. Bezzateev, Dr. Sc. (of Technical), Head of Information Security Department, State University of Aerospace Instrumentation, Professor of the Faculty of Information Technology Security, ITMO University, St. Petersburg, Russia. Orcid.org/00000002-0924-6221.E-mail: bsv@guap.ru

8. Fathi, V.A., D'yachenko N.V. Testirovanie bezopasnosti prilozhenij // Inzhenernyj vestnik Dona. 2021. № 5(77). pp. 108–120.
9. Kuz'mina, S.P. Rol' pajplajnov v sovremennoj kiberbezopasnosti: avtomatizaciya, zashchita i reagirovanie na ugrozy // Internauka. 2024. № 33-1(350). pp. 9-10.
10. Gorshkov A.G., Pitelinskij K.V., Mihajlov I.A., Mednikova Z.M., Umerzakova D.A. Osobennosti primeneniya koncepcii SSDLC pri razrabotke i testirovanii zashchishchennogo programmnogo obespecheniya // Informacionnye tekhnologii v proektirovanii i proizvodstve. 2024. № 3(195). pp. 30–36. DOI: 10.52190/2073-2597_2024_3_30.
11. Naidoo, R. Building Software Applications Securely with DevSecOps: A Socio-Technical Perspective // European Conference on Cyber Warfare and Security. 2022. № 1 (21). pp. 198–205. DOI: 10.34190/eccws.21.1.295.
12. Majorova, E.V., Sokolovskaya S.A., Chertok A.V. Obespechenie informacionnoj bezopasnosti pri gibkom podhode razrabotki programmnogo produkta // Cifrovye tekhnologii obrabotki i zashchity informacii: Sbornik nauchnyh statej / Pod redakciej E.V. Stel'mashonok, I.N. Vasil'evoj. Saint-Peterburg: Sankt-Peterburgskij gosudarstvennyj ekonomicheskij universitet, 2020. pp. 83–92.