**БЕЗОПАСНОСТЬ ЦИФРОВЫХ ТЕХНОЛОГИЙ. – 2025. – № 3 (118).** – 62–73

*МЕТОДЫ И СИСТЕМЫ ЗАЩИТЫ ИНФОРМАЦИИ, ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ*

УДК 004 DOI: 10.17212/2782-2230-2025-3-62-73

# ПРАКТИКА БЕЗОПАСНОЙ РАЗРАБОТКИ ПРИ СОЗДАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ АВТОМАТИЗИРОВАННЫХ СТЕНДОВ*

Р.Е. КАЧУРА[1], И.Д. КАРПОВ[2], С.А. ЗЫРЯНОВ[3]

[1] *630082, РФ, г. Новосибирск, ул. Дачная, 60/1, АО «Радио и микроэлектроника», инженер-программист. E-mail: kachura@zao-rim.ru*
[2] *630082, РФ, г. Новосибирск, ул. Дачная,60/1, АО «Радио и микроэлектроника», заместитель начальника СКБ. E-mail: karpov@zao-rim.ru*
[3] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технических университет, кандидат технических наук, доцент кафедры защиты информации. E-mail: zyryanov@corp.nstu.ru*

Повышенные требования к точности и надежности измерительных систем обусловливают особые требования к информационной безопасности программного обеспечения автоматизированных стендов. В работе рассматриваются практики безопасной разработки, включая внедрение защитных механизмов на этапе проектирования, использование статического анализа кода, а также контроль уязвимостей на всех этапах жизненного цикла разработки. Особое внимание уделено интеграции мер безопасности в архитектуру программного обеспечения и процесс его тестирования. Кроме того, обсуждается организация обучения разработчиков и проведение ревизий кода для поддержания постоянного уровня защищенности. Представленные подходы позволяют снизить вероятность эксплуатации уязвимостей и повысить устойчивость систем к возможным атакам и отказам. Предложенные методы могут быть внедрены при разработке специализированных программных продуктов в различных областях промышленности.

**Ключевые слова**: информационная безопасность, безопасная разработка, автоматизированные стенды, программное обеспечение, статический анализ, жизненный цикл разработки, уязвимости программного кода, метрологические системы

## ВВЕДЕНИЕ

Современные программные системы, управляющие техническими и производственными процессами, всё чаще становятся объектами кибератак. Особенно уязвимыми оказываются специализированные программные про-

* Статья получена 01 июля 2025 г.

дукты, разрабатываемые под конкретные задачи и не всегда проходящие полноценную проверку на соответствие требованиям информационной безопасности. Отсутствие встроенных защитных механизмов и контрольных процедур может привести к несанкционированному доступу, искажению данных или нарушению функционирования критически важного оборудования.

Разработка программного обеспечения для автоматизированных измерительных стендов сопряжена с необходимостью обеспечения надежности и устойчивости к внешним воздействиям. Однако в условиях ограниченных ресурсов и сжатых сроков часто упускаются из виду аспекты защищенности кода, архитектуры и среды исполнения. Это создает потенциальные угрозы, особенно при эксплуатации таких стендов в составе технологических комплексов, подключенных к корпоративным или производственным сетям.

Эффективным подходом к снижению числа уязвимостей на этапах проектирования и реализации является внедрение практик безопасной разработки. К ним относятся как организационные меры (разделение прав, ведение документации, аудит изменений), так и технические, в том числе применение инструментов статического анализа кода, позволяющих выявлять потенциально опасные конструкции до этапа компиляции и тестирования.

Большое значение имеет архитектурный подход, предполагающий проектирование компонентов системы с учетом принципов безопасности: минимизация прав доступа, изоляция критических модулей, контроль взаимодействия между компонентами и их внешней средой. Комплексное использование таких подходов позволяет создавать более устойчивые и предсказуемо работающие программные решения.

Целью работы является анализ и обоснование практик безопасной разработки программного обеспечения, примененных в рамках создания специализированной автоматизированной системы.

Для достижения поставленной цели были выделены следующие задачи:

- выявить особенности разработки ПО для специализированных технических стендов с точки зрения информационной безопасности;
- обосновать выбор архитектурных решений, повышающих защищенность системы;
- описать процесс внедрения статического анализатора кода и оценить его эффективность.

## 1. ОСОБЕННОСТИ РАЗРАБОТКИ ПО ДЛЯ АВТОМАТИЗИРОВАННЫХ СТЕНДОВ

Автоматизированные стенды представляют собой автоматизированные системы, предназначенные для проведения поверки измерительных датчиков. Программное обеспечение, управляющие такими стендами, играет ключевую роль в обеспечении точности, достоверности и повторяемости измерительных операций. В отличие от универсальных программных решений, подобные системы имеют жесткую привязку к аппаратной части, специфическим протоколам обмена и т. д.

Одной из особенностей разработки ПО для автоматизированных стендов является высокая чувствительность к ошибкам как в логике управления, так и в реализации алгоритмов измерения и обработки данных. Ошибки, допущенные на уровне программного обеспечения, могут привести не только к некорректной работе стенда, но и к нарушению метрологических характеристик проверяемых устройств, что ведет к серьезным последствиям на производстве и в эксплуатации.

Еще одним важным аспектом является ограниченность среды исполнения. В большинстве случаев программные комплексы работают в условиях контролируемой среды, но при этом могут быть подключены к внешним информационным системам: базам данных, серверным приложениям, сетям предприятия. Это расширяет потенциальную зону атаки и требует от разработчиков учитывать вектор возможного внешнего воздействия, включая попытки несанкционированного доступа, подмены калибровочных коэффициентов или внедрения вредоносного кода.

Важной характеристикой таких программных продуктов является длительных жизненный цикл. Программное обеспечение автоматизированного стенда, как правило, разрабатывается на годы вперед и может использоваться в стабильной конфигурации без значительных изменений. Это требует повышенного внимания к качеству кода, архитектуре системы и механизмам обновления, поскольку позднее устранение уязвимостей может быть затруднено или вовсе невозможно без остановки работы оборудования.

Таким образом, разработка такого рода программного обеспечения предъявляет к разработчику требования не только к функциональности и точности, но и к защищенности, устойчивости к сбоям и способности противостоять потенциальным угрозам, что делает применение практик безопасной разработки особенно актуальным.

## 2. АРХИТЕКТУРНЫЕ РЕШЕНИЯ

Разрабатываемое программное обеспечение представляет собой специализированное WPF-приложение для управления автоматизированным стендом, выполняющим поверку и калибровку датчиков тока и напряжения. Основные функции приложения: управление измерительным оборудованием по заданным сценариям, обработка результатов, формирование отчетной документации и взаимодействие с базой данных предприятия. Особое внимание уделялось реализации защищенного интерфейса взаимодействия с пользователем и соблюдению точности измерений при автоматизированной обработке.

Одним из ключевых факторов, влияющих на защищенность программного обеспечения, является его архитектура. Правильно спроектированная архитектура системы позволяет заложить основу для минимизации уязвимостей, упрощения контроля доступа и обеспечения устойчивости к сбоям. При создании программного обеспечения для стенда были реализованы архитектурные решения, направленные на повышение общей защищенности системы.

В качестве архитектурного шаблона был выбран MVVM (Model-View-ViewModel) – современный подход, широко применяемый в WPF-приложениях. Разделение представления, логики и данных позволило достичь модульности, изоляции компонентов и удобства сопровождения. Это также упростило реализацию контроля прав доступа на уровне ViewModel без вынесения логики в представление (пользовательский интерфейс) [1].

Схема взаимодействия между структурными частями шаблона MVVM представлена на рис. 1.

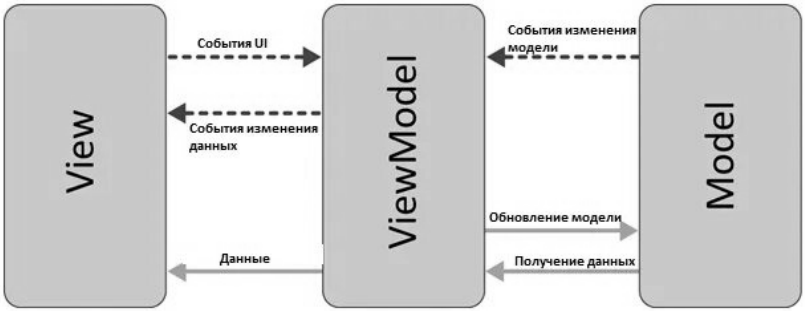


*Рис. 1*. Реализация шаблона MVVM

Важным направлением обеспечения защищенности стала интеграция с корпоративной системой управления доступом – Active Directory. Использование механизма единого входа позволило упростить аутентификацию пользователей и исключить необходимость хранения паролей внутри приложения. При входе пользователь автоматически идентифицируется на основе учетной записи в домене, после чего система определяет его роль.

В рамках проекта была реализована ролевая модель доступа, включающая следующие уровни:

- **оператор** (имеет доступ только к запуску автоматических сценариев поверочных процедур, без возможности изменения настроек);
- **инженер** (может настраивать параметры измерений, использовать ручную калибровку, а также просматривать технические логи);
- **администратор** (обладает полными правами).

Механизм авторизации был встроен в логические компоненты приложения. При инициализации сессии проверяется принадлежность пользователя к группам AD и присваивается соответствующий набор разрешений. Это позволило исключить реализацию собственного механизма управления доступом и минимизировать риски, связанные с ошибками в логике авторизации.

Отдельное внимание было уделено проектированию изолированных модулей: модуль взаимодействия с оборудованием не имеет прямого доступа к пользовательскому интерфейсу или хранилищу данных. Обмен данными осуществляется строго через контролируемые интерфейсы, реализующие валидацию и фильтрацию. Такой подход снижает риск непреднамеренного распространения ошибочных данных или выполнение несанкционированных команд.

В целях повышения устойчивости и отказоустойчивости приложения были внедрены следующие архитектурные решения:

- **обработка всех внешних исключений** с регистрацией в журнале событий, включая ошибки оборудования и сетевых подключений;
- **логирование действий пользователей** с привязкой к доменной учетной записи и временной метке;
- **механизм резервного копирования конфигурационных файлов** при изменении критических параметров;
- **защита от повторного запуска опасных операций** (например, операции с тестовыми параметрами).

Архитектура приложения также учитывает возможность масштабирования и обновления. Все настройки хранятся в отдельном конфигурационном модуле, а структура команд и сообщений спроектирована с учетом возможности расширения без изменения базовой логики.

Принятые решения обеспечили четкое разграничение прав доступа, устойчивость к ошибкам и предсказуемое поведение приложения даже при нестандартных сценариях использования.

## 3. ВНЕДРЕНИЕ ИНСТРУМЕНТОВ СТАТИЧЕСКОГО АНАЛИЗА

Статический анализ кода – один из наиболее эффективных методов выявления уязвимостей и дефектов на раннем этапе разработки программного обеспечения. В отличие от динамического анализа, он не требует запуска приложения, что позволяет выявлять ошибки, которые могут не проявляться в процессе тестирования, но способны привести к серьезным последствиям в процессе эксплуатации.

В рамках проекта была поставлена задача интеграции инструментов статического анализа в процесс разработки программного обеспечения автоматизированного стенда. В качестве основного инструмента анализа был выбран **SonarQube** – широко распространенная платформа с поддержкой большого количества языков программирования, в том числе C#, применяемого в проекте [2].

Для обеспечения автономности и соответствия требования внутренней безопасности SonarQube был развернут на локальном сервере внутри корпоративной сети. Это позволило исключить передачу исходного кода на внешние ресурсы и обеспечить соответствие внутренним политикам конфиденциальности и контроля данных.

Интеграция SonarQube с репозиторием GitHub, используемым в компании, была реализована через систему CI/CD. Каждый коммит или pull request автоматически инициировал процесс анализа кода, запуск которого происходил на локальном агенте сборки. Конвейер был реализован с использованием GitHub Actions, в конфигурации которых был прописан этап подключения к локальному серверу SonarQube через API и анализ с последующей отправкой результатов.

Пример запроса к API SonarQube показан на рис. 2.

```
curl --request GET \
  --url 'https://{АДРЕС СЕРВЕРА}/api/measures/component?metricKeys=ncloc%2Ccode_smells%2Ccomplexity&component=AKIP' \
  --header 'Authorization: Bearer {СЕКРЕТНЫЙ КЛЮЧ}'
```

*Рис. 2*. Запрос к API SonarQube, получение метрик

Автоматическая проверка позволила следующее:

- обнаруживать потенциально уязвимые участки кода, включая использование устаревших библиотек, необработанные исключения, дублирование логики и возможные утечки данных;
- отслеживать технический долг и метрики качества проекта, включая покрытие тестами, сложность методов и повторное использование кода.

Результаты анализа автоматически отображались в интерфейсе SonarQube и были доступны всем участникам проекта. Разработчики получали уведомления об ошибках, допущенных в их коммитах, что способствовало повышению культуры безопасного программирования и постепенному улучшению качества кода. Были настроены пороговые значения: если количество критических замечаний превышало установленный лимит, то CI-конвейер отклонял сборку, не позволяя внести уязвимый код в основную ветку репозитория.

По результатам внедрения статического анализа было отмечено снижение количества ошибок, выявляемых на поздних этапах тестирования, и рост общего уровня безопасности приложения. Благодаря автоматизации процесса проверки удалось существенно повысить прозрачность разработки и упростить контроль соответствия проекта внутренним требованиям.

Для иллюстрации достигнутого эффекта на рис. 3 представлены сравнительные данные по выявленным критическим уязвимостям и потенциальным ошибкам до и после внедрения системы статического анализа SonarQube.

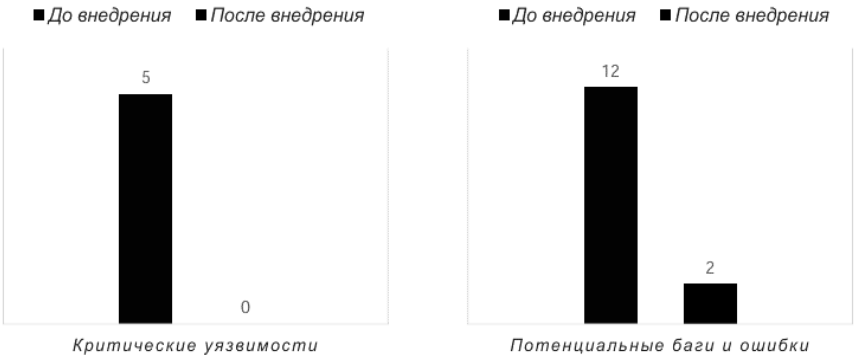


*Рис. 3.* Влияние внедрения статического анализа на качество кода

## 4. ПРАКТИЧЕСКИЕ АСПЕКТЫ БЕЗОПАСНОЙ РАЗРАБОТКИ

Безопасная архитектура и статический анализ кода являются важными компонентами защищенного программного продукта, однако они должны дополняться рядом практических мер, направленных на обеспечение безопасности как на этапе разработки, так и в процессе эксплуатации программного обеспечения. В рамках проекта эти меры реализовывались системно, с учетом специфики стенда и требований к точности и устойчивости.

Одной из ключевых практик стала реализация разграничения прав доступа. Как уже отмечалось ранее, авторизация пользователей происходила через Active Directory, что обеспечивало единый механизм управления доступом. Дополнительно на уровне логики приложения было внедрено гибкое управление ролями, при котором каждая операция проверялась на наличие соответствующих прав. Такой подход позволил исключить выполнение критических действий (например, изменение конфигурации оборудования или редактирование шаблонов поверки) пользователями без соответствующих полномочий.

Для повышения заметности и возможности последующего анализа действий пользователей был реализован механизм централизованного логирования. В журнал записывались ключевые события:

- вход и выход пользователей;
- запуск, отмена и завершение ключевых операций;
- изменение конфигурации или калибровочных параметров;
- попытки выполнения недопустимых действий.

Каждое событие содержало информацию о времени, имени пользователя (на основе данных из Active Directory) и деталях операции. Это позволило обеспечить аудит действий в системе и основу для проведения внутренних расследований при подозрении на несанкционированную активность.

Отдельное внимание было уделено обеспечению безопасности хранения и обработки конфигурационных данных. Все критически важные параметры сохранялись в защищенном формате и дублировались в резервные копии. При каждом изменении создавалась версия с возможностью восстановления предыдущего состояния, что исключало необратимую потерю данных. В процессе разработки применялись внутренние регламенты и код-стайл, направленные на унификацию решений и предотвращение типичных ошибок.

Разработчики проходили инструктаж по безопасному программированию по следующим темам:

- защита от SQL-инъекций и небезопасного обращения с данными;
- корректная обработка исключений и логирование ошибок;
- безопасное взаимодействие с внешними библиотеками.

Кроме того, в проектной документации были отражены политики обновления и сопровождения: регламент тестирования новых версий, проверки обратной совместимости и условий вывода обновлений в эксплуатацию. Это снизило риски, связанные с внедрением новых версий ПО, и дало возможность оперативно устранить найденные уязвимости.

Принятие комплекса практических мер позволило не только реализовать надежную защиту внутри системы, но и создать условия для ее устойчивой и безопасной эксплуатации в течение всего жизненного цикла. Применение этих подходов подтверждает, что обеспечение информационной безопасности является не отдельным этапом, а неотъемлемой частью современного процесса разработки программных решений.

## ЗАКЛЮЧЕНИЕ

Практика безопасной разработки является важнейшим условием при создании специализированного программного обеспечения, особенно в системах, где требуется высокая точность, надежность и предсказуемость. Применение архитектурных решений на основе шаблона MVVM, интеграции с корпоративной системой аутентификации (Active Directory), а также внедрение ролевой модели доступа позволило сформировать надежную структуру защиты на уровне архитектуры приложения.

Дополнительный уровень контроля обеспечил развернутый на локальном сервере инструмент статического анализа SonarQube, интегрированный с CI/CD в GitHub. Автоматический анализ каждого изменения в коде позволил снизить количество критических уязвимостей и повысить общее качество программного продукта.

Также были внедрены меры по логированию, аудиту действий пользователей, резервному копированию данных и регламентированному обновлению. Эти шаги обеспечили не только защиту от внешних и внутренних угроз, но и повышенную устойчивость к сбоям и ошибкам эксплуатации.

Представленные в работе решения могут быть масштабированы и адаптированы для других проектов, где требуется сочетание надежности, точности и информационной безопасности. Таким образом, обеспечение безопасности стало не дополнительной функцией, а встроенной частью процесса проектирования и разработки программного обеспечения. Выявленные проблемы и способы их устранения демонстрируют применимость безопасной разработки в условиях ограниченного ресурса и отраслевой специфики.

## СПИСОК ЛИТЕРАТУРЫ

1. Model – View-ViewModel (MVVM) // Microsoft: сайт. – 10.09.2024. – URL: https://learn.microsoft.com/ru-ru/dotnet/architecture/maui/mvvm (дата обращения: 03.09.2025).

2. SonarQube Server // SonarSource: website. – 2025. – URL: https://www.sonarsource.com/products/sonarqube/ (accessed: 03.09.2025).

3. Переход к безопасной разработке. Зачем это нужно? Какие преимущества даст DevSecOps? // Habr: сайт. – 2022. – URL: https://habr.com/ru/articles/661347/ (дата обращения: 03.09.2025).

4. Документация GitHub Actions. – 2025. – URL: https://docs.github.com/ru/actions (дата обращения: 03.09.2025).

5. *Черемисин Д.Г., Мкртчян В.Р.* Безопасная разработка программного обеспечения // Символ науки. – 2023. – № 6-2. – URL: https://cyberleninka.ru/article/n/bezopasnaya-razrabotka-programmnogo-obespecheniya (дата обращения: 03.09.2025).

6. Безопасная разработка (SSDLC, DevSecOpc) // Cisoclub. – 02.04.2024. – URL: https://cisoclub.ru/bezopasnaja-razrabotka-ssdlc-devsecops/ (дата обращения: 03.09.2025).

7. Безопасная разработка: обзор основных инструментов // Habr: сайт. – 2024, 12 апреля. – URL: https://habr.com/ru/companies/yandex_praktikum/articles/807053/ (дата обращения: 03.09.2025).

8. *Алексеев А.Л.* Эволюция и анализ моделей жизненного цикла разработки программного обеспечения // Вестник науки. – 2024. – № 7 (76). – URL: https://cyberleninka.ru/article/n/evolyutsiya-i-analiz-modeley-zhiznennogo-tsikla-razrabotki-programmnogo-obespecheniya (дата обращения: 03.09.2025).

9. Общие сведения о доменных службах Active Directory // Microsoft: сайт. – 16.08.2025. – URL: https://learn.microsoft.com/ru-ru/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview (дата обращения: 03.09.2025).

10. Безопасная разработка: как обеспечить безопасность разрабатываемых продуктов // Habr: сайт. – 2024, 1 ноября. – URL: https://habr.com/ru/articles/855276/ (дата обращения: 03.09.2025).

***Качура Ростислав Евгеньевич***, инженер-программист, АО «Радио и микроэлектроника». Область научных интересов – информационная безопасность, безопасная разработка. E-mail: kachura@zao-rim.ru

***Карпов Игорь Дмитриевич***, заместитель начальника СКБ, АО «Радио и микроэлектроника». Область научных интересов – информационная безопасность, компьютерные системы. E-mail: karpov@zao-rim.ru

***Зырянов Сергей Алексеевич,*** кандидат технических наук, доцент кафедры защиты информации Новосибирского государственного технического университета. Основное направление научных исследований – технологии построения информационно-телекоммуникационных сетей. E-mail: zyryanov@corp.nstu.ru

DOI: 10.17212/2782-2230-2025-3-62-73

## Secure development practice in the creation of software for automated test stands *

**R.E. Kachura[1], I.D. Karpov[2], S.A. Zyryanov[3]**

[1] *JSC "Radio and Microelectronics", 60/1 Dachnaya Street, Novosibirsk, 630082, Russian Federation, Software Engineer. E-mail: kachura@zao-rim.ru*
[2] *JSC "Radio and Microelectronics", 60/1 Dachnaya Street, Novosibirsk, 630082, Russian Federation, Deputy Head of Design Bureau. E-mail: karpov@zao-rim.ru*
[3] *Novosibirsk State Technical University, 20 Karl Marx Prospekt, Novosibirsk, 630073, Russian Federation, Ph. D. in Engineering, Associate Professor of the Department of Information Security. E-mail: zyryanov@corp.nstu.ru*

Modern requirements for the accuracy and reliability of measurement systems place increased demands on the information security of software used in verification stands. This paper examines secure development practices, including the implementation of protective mechanisms at the design stage, static code analysis, and vulnerability control throughout the entire software development life cycle. Special attention is given to integrating security measures into software architecture and the testing process. The paper also discusses developer training and code review procedures aimed at maintaining a consistent level of security. The approaches presented help reduce the likelihood of vulnerability exploitation and increase the resilience of systems to potential attacks and failures. The proposed methods can be applied in the development of specialized software solutions across various industrial domains.

**Keywords**: information security, secure development, verification stands, software, static analysis, software development life cycle, code vulnerabilities, measurement systems

## REFERENCES

1. Model – View-ViewModel (MVVM). *Microsoft*. Website. 10.09.2024. Available at: https://learn.microsoft.com/ru-ru/dotnet/architecture/maui/mvvm (accessed 03.09.2025).

2. SonarQube Server. *SonarSource*. Website. 2025. Available at: https://www.sonarsource.com/products/sonarqube/ (accessed 03.09.2025).

* *Received 01 July 2025.*

3. Habr. *Perekhod k bezopasnoi razrabotke. Zachem eto nuzhno? Kakie preimushchestva dast DevSecOps?* [Transition to secure development. Why is this needed? What are the benefits of DevSecOps?]. (In Russian). Available: https://habr.com/ru/articles/661347/ (accessed 03.09.2025).

4. GitHub. *GitHub Actions documentation*. (In Russian). Available: https://docs.github.com/ru/actions (accessed 03.09.2025).

5. Cheremisin D.G., Mkrtchyan V.R. Bezopasnaya razrabotka programmnogo obespecheniya [Secure software development]. *Simvol nauki = Symbol of Science*, 2023, no. 6-2. Available: https://cyberleninka.ru/article/n/bezopasnaya-razrabotka-programmnogo-obespecheniya (accessed 03.09.2025).

6. Bezopasnaya razrabotka (SSDLC, DevSecOpc) [Secure development (SSDLC, DevSecOps)]. *Cisoclub.ru*, 02.04.2024. Available at: https://cisoclub.ru/bezopasnaja-razrabotka-ssdlc-devsecops/ (accessed 03.09.2025).

7. Habr. *Bezopasnaya razrabotka: obzor osnovnykh instrumentov* [Secure development: overview of key tools]. 2024, April 12. Available: https://habr.com/ru/companies/yandex_praktikum/articles/807053/ (accessed 03.09.2025).

8. Alekseev A.L. Evolyutsiya i analiz modelei zhiznennogo tsikla razrabotki programmnogo obespecheniya [Evolution and analysis of software development life cycle models]. *Vestnik nauki = Bulletin of Science*, 2024, no. 7 (76). Available: https://cyberleninka.ru/article/n/evolyutsiya-i-analiz-modeley-zhiznennogo-tsikla-razrabotki-programmnogo-obespecheniya (accessed 03.09.2025).

9. Microsoft. *Obshchie svedeniya o domennykh sluzhbakh Active Directory* [Active Directory Domain Services overview]. Available: https://learn.microsoft.com/ru-ru/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview (accessed 03.09.2025).

10. Habr. *Bezopasnaya razrabotka: kak obespechit' bezopasnost' razrabatyvaemykh produktov* [Secure вevelopment: How to ensure the security of the products you вevelop]. Available: https://habr.com/ru/articles/855276/ (accessed 03.09.2025).

Для цитирования:

*Качура Р.Е., Карпов И.Д., Зырянов С.А.* Практика безопасной разработки при создании программного обеспечения для автоматизированных стендов // Безопасность цифровых технологий. – 2025. – № 3 (118). – С. 62–73. – DOI: 10.17212/2782-2230-2025-3-62-73.

For citation:

Kachura R.E., Karpov I.D., Zyryanov S.A. Praktika bezopasnoi razrabotki pri sozdanii programmnogo obespecheniya dlya avtomatizirovannykh stendov [Secure development practice in the creation of software for automated test stands]. *Bezopasnost' tsifrovykh tekhnologii = Digital Technology Security*, 2025, no. 3 (118), pp. 62–73. DOI: 10.17212/2782-2230-2025-3-62-73.