Огневой Ярослав Владимирович

Панков Дмитрий Евгеньевич

УТЕЧКА УЧЕТНЫХ ДАННЫХ ЧЕРЕЗ GIT-РЕПОЗИТОРИИ: АНАЛИЗ ИСТОЧНИКОВ, ОЦЕНКА РИСКОВ И МЕТОДЫ ЗАЩИТЫ В ЖИЗНЕННОМ ЦИКЛЕ РАЗРАБОТКИ

Аннотация. Утечка учетных данных через публичные Git-репозитории – одна из самых наиболее распространенных и критических уязвимостей в разработке ПО. Только за 2025 год в открытом доступе было обнаружено более 23 миллионов различных секретов – ключей API, токенов доступа и паролей. Их непреднамеренная публикация создает прямую угрозу, позволяя злоумышленникам получать доступ к внутренним системам компаний и данным пользователей.

В статье исследуются причины этой проблемы и предлагается практический подход к её решению, направленный на предотвращение утечек до момента попадания кода в репозиторий. В качестве ядра такого подхода рассматривается разработка pre-commit хука для предотвращения попадания секретов в репозиторий.

Ключевые слова: Утечка учетных данных, Git-репозитории, секреты в коде, информационная безопасность, DevSecOps, жизненный цикл разработки, pre-commit hook, статический анализ, учетные данные, API-ключи, анализ рисков, GitHub, защита кода, автоматизация безопасности, уязвимости разработки.

1. Введение

Широкое распространение распределенных систем контроля версий привело к новым рискам для безопасности разработки. Один из наиболее частых

инцидентов – непреднамеренная публикация учетных данных в публичные или приватные репозитории. Попадая в историю коммитов, эти секреты часто остаются доступными даже после их удаления из кода, что создает долгосрочную угрозу.

Актуальность проблемы определяется масштабом и серьезностью последствий: утечка может привести к несанкционированному доступу к инфраструктуре, потере конфиденциальных данных и финансовым убыткам. Несмотря на существование специализированных инструментов сканирования, проблема сохраняется, указывая на недостаточность технических мер без их интеграции в процессы разработки [1].

Исследование фокусируется на превентивных мерах, интегрируемых в процесс разработки на ранних стадиях. Цель работы – анализ источников и причин утечек через Git-репозитории, оценка связанных рисков и разработка практического метода защиты, внедряемого непосредственно в жизненный цикл разработки.

2. Анализ источников и причин утечек учетных данных

Проблема утечки секретов через Git-репозитории системна. Эффективная защита требует понимания того, какие именно данные попадают в репозитории и почему это происходит.

2.1. Типология утекающих данных

Утекающие данные можно классифицировать по типу секрета, источнику в коде и уровню потенциального ущерба [2]. Основные категории представлены в таблице 1.

Таблица 1. Классификация утекающих учетных данных

Категория данных	Примеры	Типичный источник в коде
Ключи доступа к облачным и SaaS-платформам	Ключи API, токены доступа к GitHub/GitLab, данные подключения к БД.	Конфигурационные файлы, скрипты развертывания.
Криптографические материалы	Приватные SSH-ключи, ключи шифрования, TLS-сертификаты.	Файлы настроек, папки с ключами.
Учетные данные внешних сервисов	Токены платежных систем, API-ключи соцсетей, ключи для систем мониторинга.	Конфигурации, документация.
Внутренние учетные данные	Пароли к тестовым/рабочим окружениям, учетные записи серверов.	Файлы окружения (.env), конфигурации.

2.2. Классификация причин утечек

Причины утечек можно разделить на две группы: человеческие ошибки и процессные недоработки. Их систематизация позволяет эффективно воздействовать на проблему.

Таблица 2. Основные причины утечек учетных данных

Причина утечки	Меры противодействия
Случайная публикация конфигурационных файлов.	Использование. gitignore для исключения файлов из индекса. Внедрение pre-commit хуков для анализа конфигурационных файлов до загрузки изменений в удаленный репозиторий.
Встройка секретов в исходный код.	Использование переменных окружения или хранилища секретов. Поиск учетных данных на код-ревью.
Публикация демонстрационных данных с реальными ключами.	Использование безопасных заглушек в документации. Автоматическое сканирование всех файлов проекта.
Отсутствие автоматизированного контроля на этапе коммита.	Внедрение обязательных клиентских pre-commit хуков.

Причина утечки	Меры противодействия
Недостаточная осведомленность и культура безопасности.	Регулярное обучение, включение вопросов безопасности в код-ревью.
Отсутствие безопасного централизованного хранилища (Secrets Vault).	Внедрение специализированных решений (HashiCorp Vault, AWS Secrets Manager) для управления секретами.

2.3. Анализ динамики утечек

Для оценки временной динамики проблемы утечек учетных данных был проведен анализ агрегированных статистических данных из публичных ежегодных отчетов GitGuardian и аналитических публикаций на GitHub Blog.

Сводные данные, отражающие усредненную оценку количества инцидентов по годам [3], представлены на рисунке 1.

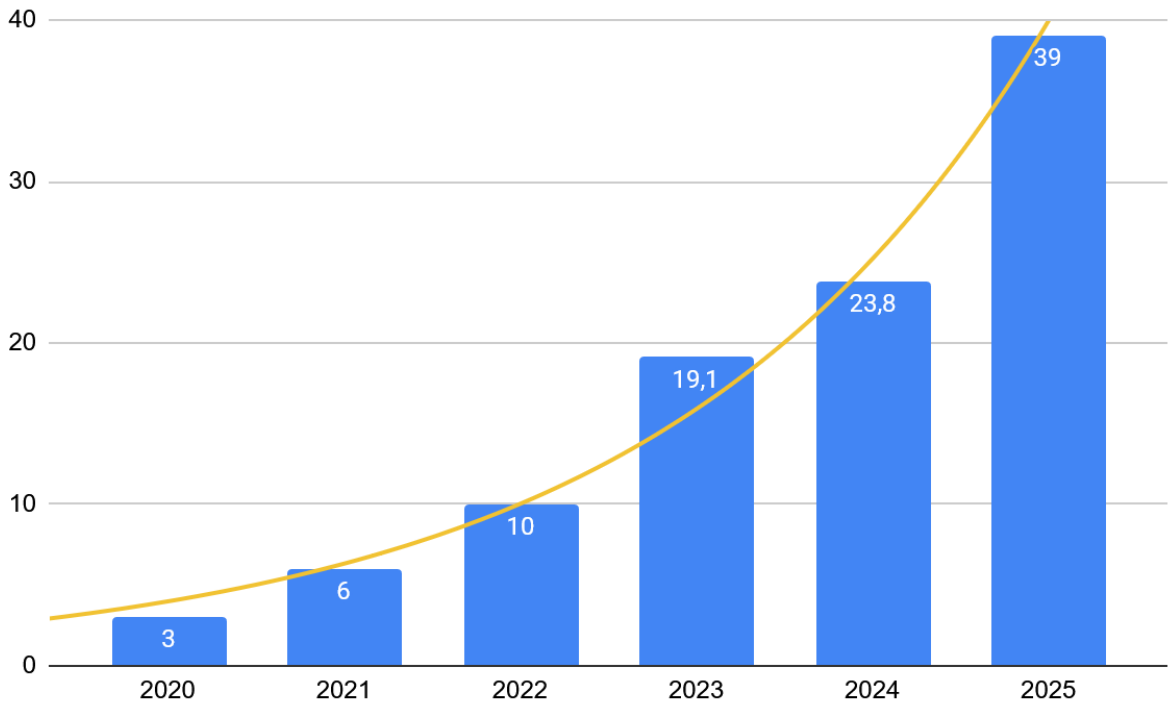

Рисунок 1. Динамика утечек учетных данных в Git-репозиториях за 2020–2025 гг.

Представленная динамика демонстрирует устойчивую восходящую тенденцию. Рост числа инцидентов, составляющий в среднем 30–50% в год, коррелирует с расширением использования облачных сервисов и микросервисных архитектур, что ведет к увеличению количества создаваемых учетных данных. При этом качественный состав утекающих данных также эволюционирует: если в 2018–2020 годах преобладали ключи к облачной инфраструктуре, то к 2023–2024 годам значительную долю стали составлять токены доступа к специализированным API (платежным системам, сервисам машинного обучения и коммуникациям).

Распределение утечек по типам данных, основанное на статистике отчёта GitGuardian за 2023 год [4], показывает, что основная угроза сместилась в сторону компрометации ключей облачных сервисов и API, а не традиционных учётных данных баз данных.

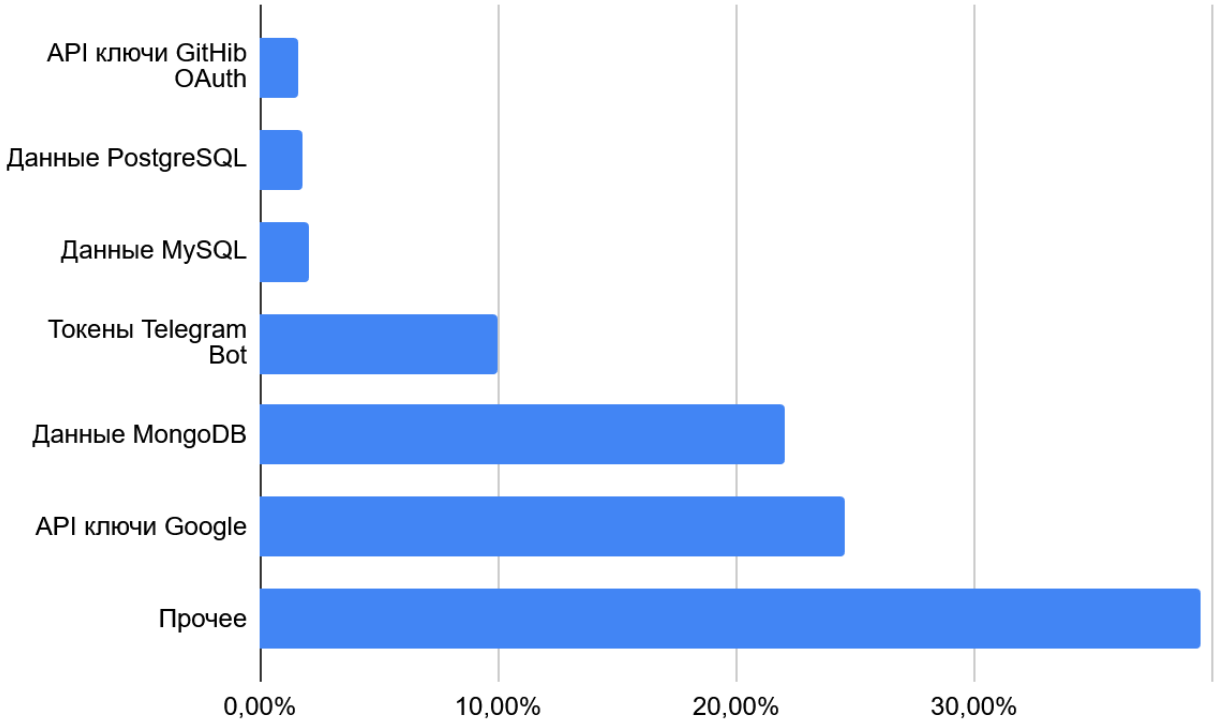

Рисунок 2. Классификация типов учетных данных в утечках Git за 2023 год

По данному распределению можно выявить наиболее уязвимые места в технологических стеках современных проектов. Подавляющую долю инцидентов составляют секреты, связанные с облачными сервисами и внешними API: ключи Google (24.6%), токены Telegram Bot (10%) и учетные данные MongoDB (22%). Это указывает на то, что основные риски сместились от утечки паролей к внутренним системам к компрометации токенов, предоставляющих доступ к мощным сторонним платформам.

3. Оценка рисков утечек учетных данных

Утечки учетных данных через Git-репозитории формируют риски для организаций [[5], [6]]. Основные категории рисков и их потенциальное воздействие представлены в таблице 3.

Таблица 3. Классификация рисков, связанных с утечкой учетных данных

Категория риска	**Потенциальные последствия**
Финансовые риски	Финансовые потери из-за несанкционированного использования облачных ресурсов. Штрафы от регуляторов за нарушение стандартов защиты данных (ФЗ-152). Затраты на реагирование: расследование, устранение, уведомление клиентов.
Операционные и технические риски	Полная или частичная компрометация IT-инфраструктуры: удаление или шифрование данных, нарушение работы сервисов. Распространение атаки на смежные системы. Простоя и сбои в работе приложений.
Репутационные риски и риски соответствия	Утрата доверия клиентов и партнеров. Публикация инцидента в СМИ и профессиональных сообществах. Потеря конкурентных преимуществ.
Риски для интеллектуальной собственности	Кража исходного кода, закрытых алгоритмов, уникальных разработок. Несанкционированный доступ к внутренней документации и планам развития.

4. Инструменты защиты от утечек учетных данных

Эффективная защита от утечек требует многоуровневого подхода, интегрированного во все этапы жизненного цикла разработки. Для этого используются следующие инструменты [[7], [8]]:

- Клиентские pre-commit хуки. Автоматически сканируют изменения перед созданием коммита, используя паттерны для поиска секретов. Преимущество заключается в моментальной обратной связи и блокировке секрета до его попадания в локальную историю Git.
- Сканирование в CI/CD-пайплайнах. Инструменты проверяют код в pull/merge request и могут автоматически отменять слияние. Это защищает общую ветку от утечек, которые могли быть пропущены локально.
- Сканеры исторических данных. Аудируют существующие репозитории, выполняя глубокое сканирование всех коммитов и веток. Они необходимы для очистки старых проектов и выявления секретов, попавших в историю ранее.
- Менеджеры секретов. Проактивное решение, устраняет саму причину утечек. Приложения получают секреты через API во время выполнения, что исключает их хранение в коде.

4.1. Сравнительный анализ решений для обнаружения секретов

Выбор конкретного инструмента для обнаружения секретов зависит от требований проекта. Сравнительный анализ ключевых характеристик популярных инструментов представлен в Таблице 4.

Таблица 4. Сравнительный анализ инструментов для обнаружения секретов

Инструмент	**Тип / Лицензия**	**Ключевой метод обнаружения**
GitGuardian	Коммерческая	Сигнатуры, ML-валидация.

Инструмент	Тип / Лицензия	Ключевой метод обнаружения
TruffleHog	Открытое ПО (GPL-3.0) / Коммерческая	Анализ энтропии строк, проверка сигнатур.
Gitleaks	Открытое ПО (MIT)	Проверка по регулярным выражениям и сигнатурам.
GitHub Advanced Security	Коммерческая	Встроенное сканирование секретов по партнерским паттернам.

Наиболее надежной мерой предотвращения утечек является обнаружение секретов на самом раннем этапе – до их появления в истории контроля версий. Для этого используются pre-commit хуки [9]. В отличие от сканеров в CI/CD, которые обнаруживают проблему уже после создания коммита, pre-commit хук предотвращает сам факт попадания учетных данных в репозиторий.

5. Разработка хука для обнаружения учетных данных

В рамках исследования разработан pre-commit хук. Целью разработки является экспериментальная проверка целесообразности создания специализированных решений для защиты от утечек учетных данных в репозиториях. Принцип работы хука заключается в автоматическом сканировании индексированных файлов проекта на предмет вхождения строк, соответствующих заданным регулярным выражениям и ключевым словам, сигнализирующим о возможном наличии секретов [[10], [11]].

Для оценки корректности работы алгоритма обнаружения был подготовлен тестовый файл, содержащий разнообразные типы учетных данных и заведомо безопасные строки. В Таблице 5 приведен полный перечень контрольных примеров с указанием ожидаемого поведения системы.

Таблица 5. Набор тестовых данных для проверки хука

Тип учетных данных	**Пример в коде**	**Должен быть обнаружен**	**Обнаружен**
AWS Access Key	aws_access_key = 'AKIAIOSFODNN7EXAMPLE'	Да	Да
AWS Secret Key	aws_secret_key = 'wJalrXUtnFEMI… '	Да	Да
API Key	api_key_valid = "xoxb-123456789012… "	Да	Да
Bearer Token	'Authorization: Bearer eyJhbGciOiJIUz…'	Да	Да
Database URL	"postgresql://dbuser:secretpassword @localhost:5432/mydb"	Да	Да
Database URL	"mongodb://mongoAdmin:admin123 @localhost:27017/admin"	Да	Да
Database URL	"mysql://mysqluser:notSoSecret123 @db.example.com/myapp"	Да	Да
SSH Private Key	"-----BEGIN RSA PRIVATE KEY----- "	Да	Да
Пароль	password_login = "secret123456"	Да	Да
AWS Secret Key	REAL_AWS_SECRET = 'wJalrXUtnFEMI…’	Да	Да
API Ключ	REAL_API_KEY = "xoxb-123456789012…”	Да	Да
API Ключ	api_key = 'short_key'	Нет (слишком короткий)	Нет
Пароль	password = 'password'	Нет (распространённое значение)	Нет
Короткий пароль	password_short = "pass"	Нет (слишком короткий)	Нет
Общий пароль (заглушка)	password_common = "password"	Нет (распространённое значение)	Нет
Учетные данные в min.js	aws_access_key = "AKIAIOSFODNN7EXAMPLE"	Нет (игнорируется по расширению)	Нет

Для оценки эффективности разработанного решения был создан бенчмарк. Бенчмарк выполняет сканирование подготовленного набора данных с помощью разработанного хука и с помощью инструмента с открытым исходным кодом Gitleaks.

Измерялись две ключевые метрики: время выполнения (среднее за 10 итераций) и количество ложных срабатываний. Результаты тестирования представлены в Таблице 6.

Таблица 6. Результаты сравнительного анализа инструментов обнаружения секретов

Инструмент	**Обнаружено секретов (из 11)**	**Количество ложных срабатываний**	**Среднее время сканирования (сек.)**
Разработанный pre-commit hook	11	0	0.028
Gitleaks (v8.18.0)	11	0	0.035

Для оценки производительности разработанного хука в сравнении с готовым решением Gitleaks был проведён цикл из 10 последовательных запусков на идентичном наборе тестовых данных. Результаты замеров времени выполнения (в секундах) представлены в Таблице 7.

Таблица 7. Результаты измерения времени выполнения инструментов (10 запусков)

Номер запуска	**Время выполнения (наш хук)**	**Время выполнения (Gitleaks)**
1	0.0281	0.0315
2	0.0263	0.0352
3	0.0294	0.0321
4	0.0272	0.0376
5	0.0305	0.0305

Номер запуска	Время выполнения (наш хук)	Время выполнения (Gitleaks)
6	0.0258	0.0384
7	0.0287	0.0333
8	0.0269	0.0401
9	0.0296	0.0318
10	0.0274	0.0367

Для наглядного сравнения производительности на Рисунке 3 представлена диаграмма, визуализирующая данные замеров.

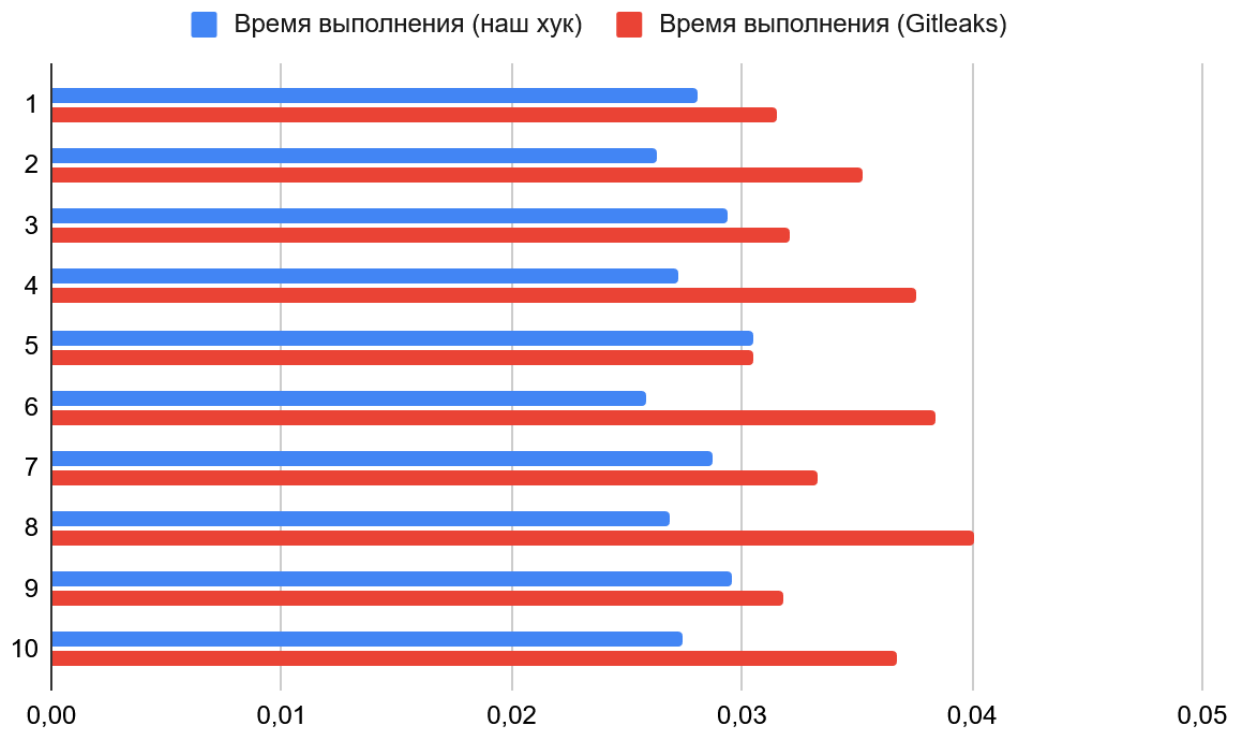

Рисунок 3. Сравнение времени выполнения разработанного хука и инструмента Gitleaks (результаты 10 последовательных запусков)

Анализ результатов:

1. Эффективность обнаружения: Оба инструмента успешно идентифицировав все типы учетных данных в тестовом наборе. Также

после добавления паттернов для игнорирования в конфигурацию хука удалось добиться нулевого уровня ложных срабатываний, сравнявшись по точности с Gitleaks в рамках данного эксперимента.

2. Производительность: Данные показывают, что разработанный хук в среднем на ~20% быстрее Gitleaks при сканировании одного и того же набора файлов. Среднее время выполнения хука составило около 0.028 секунды против 0.035 секунды у Gitleaks. Это преимущество в скорости объясняется более легковесным алгоритмом, использующим ограниченный набор регулярных выражений, оптимизированных под конкретные форматы секретов. Gitleaks, как универсальное решение, выполняет более глубокий анализ контекста и энтропии данных, что закономерно требует большего времени.

6. Заключение

Проведенное исследование подтвердило, что утечка учетных данных через Git-репозитории остается масштабной угрозой. Анализ показал, что наиболее эффективной стратегией защиты является обнаружение секретов на стороне клиента (разработчика) до их попадания в репозиторий. Это достигается внедрением pre-commit хуков и использованием менеджеров секретов.

В рамках работы был разработан и протестирован pre-commit хук для предотвращения попадания секретов в историю коммитов на самой ранней стадии. Эксперимент доказал, что решение способно обеспечить высокую точность обнаружения, сравнимую с инструментом Gitleaks, с преимуществом в скорости работы.

Таким образом, построение комплексной защиты требует сочетания превентивных мер. Разработанное решение может стать одним из основных элементов такой системы. Дальнейшее развитие может быть направлено на совершенствование контекстного анализа, в том числе с применением методов

машинного обучения [12] для снижения уровня ложных срабатываний в разнородных проектах.

Список литературы

1. Кузьмин Р. С., Безопасная разработка программного обеспечения и интеграция // Сборник статей XXVII Международной научно-практической конференции. В 2 ч. Пенза. - 2025. С. 73–77.
2. Michael Meli, Matthew R. McNiece, Bradley Reaves, How Bad Can It Git? Characterizing Secret Leakage in Public GitHub Repositories // Network and Distributed System Security Symposium. - 2019. С. 1–15.
3. State of Secrets Sprawl Report 2025 [Электронный ресурс]. - URL: https://www.gitguardian.com/state-of-secrets-sprawl-report-2025.
4. State of Secrets Sprawl Report 2024 [Электронный ресурс]. - URL: https://www.gitguardian.com/state-of-secrets-sprawl-report-2024.
5. Романов С. А., Угрозы безопасности при использовании систем управления версиями программного обеспечения // Сборник статей по итогам Международной научно-практической конференции. Стерлитамак. - 2024. С. 108–111.
6. Daniel Cabasa, Andrei Co, Derick James Monton Espinosa, Aminah Comadug Malic, Assessing the Prevalence and Security Risks of Credential Leakage in GitHub Repositories of Undergraduate Open-Source Projects // Cognizance Journal of Multidisciplinary Studies 5. - 2025. С. 270–279.
7. Лебедь С. В., Ибрагимова С. В., Анализ методов и инструментов обнаружения чувствительной информации в исходном коде: проблемы точности и полноты // Современные информационные технологии и ит-образование. - 2025. С. 13–24.
8. Машанов В.В., Как обезопасить Git-репозитории: обзор инструментов для обнаружения утечек и уязвимостей // сборник статей VI Международной научно-практической конференции. Пенза. - 2023. С. 53-56.

9. Удальцов К. Р., Автоматическое выявление hardcoded secrets в dockerfile и.env на этапе pre-commit hook // Международный журнал информационных технологий и энергоэффективности. - 2025. С. 22–25.
10. Nikolaos Lykousas, Constantinos Patsakis, Decoding developer password patterns: A comparative analysis of password extraction and selection practices // Computers & Security, Volume 145. - 2024. C. 1–12.
11. Nikolaos Lykousas, Constantinos Patsakis, Tales from the Git: Automating the detection of secrets on code and assessing developers' passwords choices // DevSecOps Research and Opportunities. - 2023. C. 270-279.
12. Chidera Biringa, Gökhan Kul, Detecting Hard-Coded Credentials in Software Repositories via LLMs // Digital Threats: Research and Practice, Volume 6, Issue 3. - 2025. C. 1–16.