УДК 004

Бадертдинова Елена Радитовна, доктор технических наук, ФГБОУ ВО «Казанский национальный исследовательский технологический университет», г. Казань

Совгачев Александр Андреевич, магистрант, ФГБОУ ВО «Казанский национальный исследовательский технологический университет», г. Казань

Краева Екатерина Владимировна, магистрант, ФГБОУ ВО «Казанский национальный исследовательский технологический университет», г. Казань

ВАЙБ-КОДИНГ: СОВРЕМЕННОЕ СОСТОЯНИЕ, ПРЕИМУЩЕСТВА И РИСКИ БЕЗОПАСНОСТИ ГЕНЕРИРУЕМОГО БОЛЬШИМИ ЯЗЫКОВЫМИ МОДЕЛЯМИ ПРОГРАММНОГО КОДА

Аннотация

В статье рассматривается новая парадигма разработки программного обеспечения — вайб-кодинг (vibe coding), введённая в 2025 г. Андреем Карпатхи. Показано, что переход к генерации кода на естественном языке обеспечивает рост производительности разработчиков до 55 % и используется в качестве основной технологии уже в 25 % стартапов Y Combinator. Вместе с тем проведён анализ, который показывает, что код, генерируемый современными большими языковыми моделями, содержит систематические уязвимости безопасности (SQL-инъекции, XSS, использование устаревших зависимостей, жёстко прописанные секреты и др.). Предложены классификация рисков, сравнение встроенных механизмов безопасности ведущих инструментов 2025 года и комплекс рекомендаций по безопасному внедрению вайб-кодинга в организациях. Особое внимание уделено стратегиям безопасного промпт-инжиниринга, включая шаблоны для генерации кода с встроенными механизмами защиты.

Annotation

The paper examines a new software development paradigm — vibe coding, introduced by Andrej Karpathy in February 2025. It is shown that the transition to

natural-language code generation provides up to 55 % developer productivity increase and is already used as a core technology in 25 % of Y Combinator startups (Winter 2025 cohort). At the same time, the analysis demonstrates that code generated by modern large language models contains systematic security vulnerabilities (SQL injection, XSS, deprecated dependencies, hard-coded secrets, etc.). A classification of risks is proposed, a comparison of built-in security mechanisms of leading 2025 tools is given, and a set of recommendations for the safe implementation of vibe coding in organizations is formulated. Special attention is paid to secure prompt engineering strategies, including templates for generating code with built-in protection mechanisms.

Ключевые слова: вайб-кодинг, генерация кода, большие языковые модели, LLM, информационная безопасность, безопасный промпт-инжиниринг, AI-assisted programming.

Keywords: vibe coding, code generation, large language models, LLM, information security, secure prompt engineering, AI-assisted programming.

В феврале 2025 года Андрей Карпатхи ввёл термин «vibe coding», обозначив им подход, при котором разработчик описывает желаемую функциональность приложения на естественном языке, а большая языковая модель (LLM) самостоятельно реализует техническую часть [12]. Этот подход радикально отличается от традиционного императивного или декларативного программирования и переводит процесс создания ПО в область «спецификации намерения» (intent-based programming).

По данным Y Combinator (внутренняя статистика Winter 2025 batch), уже 25 % стартапов используют LLM для генерации основной части кода [24]. Исследования 2025 года подтверждают рост производительности разработчиков на 43–56 % при использовании инструментов вайб-кодинга [20, 14]. Вместе с тем отчёты Veracode и OWASP фиксируют резкий рост уязвимостей в AI-генерированном коде [22, 16].

Цель настоящего обзора — систематизировать преимущества новой парадигмы, выявить основные классы рисков безопасности и предложить практически применимые меры минимизации, включая расширенный анализ безопасного промпт-инжиниринга.

Определение и эволюция вайб-кодинга

Вайб-кодинг — это итеративный процесс, в котором пользователь формулирует задачу на естественном языке, LLM генерирует исполняемый код, пользователь оценивает результат и уточняет запросы до достижения требуемой функциональности [19]. Ключевыми инструментами 2025 года являются:

- автономные агенты (Devin, OpenDevin, Aider);
- интегрированные среды (Cursor, Continue.dev, GitHub Copilot Workspace 2025);
- специализированные кодовые модели (DeepSeek-Coder-V3, Qwen-2.5-Coder-32B, Claude 4 Sonnet Technical).

По степени автономности выделяют три поколения: помощники (Copilot 2022–2024), соавторы (Cursor/Claude Projects 2024–2025) и полноценные агенты (2025+). Эволюция вайб-кодинга отражает переход от простого автодополнения к полноценным агентным системам, способным к самоанализу и рефакторингу кода [11].

Преимущества и области применения

Эмпирические исследования 2025 года показывают:

- ускорение прототипирования в 4–7 раз [11];
- рост продуктивности на 55 % при решении стандартных задач [20];
- возможность создания функциональных приложений непрограммистами (low-code/no-code на естественном языке).

Наиболее эффективно вайб-кодинг применяется для внутренних инструментов, автоматизационных скриптов, административных панелей и образовательных проектов. В отраслях, таких как финтех и здравоохранение, он позволяет быстро создавать MVP с интеграцией AI, снижая барьеры входа для доменных экспертов [23].

Причины систематической небезопасности

LLM обучаются на публичных репозиториях GitHub, содержащих значительное количество уязвимого кода (до 70 % проектов содержат хотя бы одну известную уязвимость [23]). Целевая функция обучения — автодополнение, а не обеспечение безопасности. Это приводит к воспроизведению паттернов, включая устаревшие библиотеки и отсутствие валидации [22].

Основные классы уязвимостей (данные 2025 года)

Согласно отчётам Veracode [22, 17]:

- 32 % — использование устаревших/уязвимых зависимостей;
- 27 % — SQL-инъекции и XSS;
- 19 % — жёстко прописанные секреты;
- 14 % — отсутствие валидации и санитизации входных данных;
- 8 % — race conditions и логические ошибки доступа.

Особую опасность представляют атаки prompt injection, приводящие к выполнению произвольного кода через агентные системы (LLM01:2025 по классификации OWASP) [18]. В 45 % случаев AI-генерированный код вводит OWASP Top 10 уязвимости [22].

Сравнение механизмов безопасности ведущих инструментов

Инструмент	**Основные механизмы безопасности**	**Слабые стороны**

OpenAI o3-mini-high / GPT-4o-code	Системные промпты безопасности, пост-обработка	Склонность к verbose-коду, слабый контроль зависимостей [2]
Claude 4 Sonnet Technical	Консервативный подход, встроенные пояснения	Медленнее конкурентов при больших контекстах [6]
DeepSeek-Coder-V3	Встроенный статический анализ, линтер	Слабое обнаружение логических уязвимостей [10]
GitHub Copilot Security 2025	CodeQL-интеграция, OWASP-сканирование в реальном времени	Зависимость от качества контекста репозитория [1]
Amazon CodeWhisperer + Customization	Политики соответствия, сканирование секретов	Ориентация преимущественно на AWS-экосистему [5]
Cursor Security Mode	Принудительный secure prompt template + Semgrep AI	Требует явной активации режима [21]

Таблица 1 - сравнение механизмов безопасности ведущих инструментов

Подходы к обеспечению безопасности

1. **Безопасный промпт-инжиниринг**

Эффективность доказана исследованиями 2025 года [3, 4]:

- явное указание роли («You are a senior security engineer with 15+ years…»);
- требование использования параметризованных запросов, валидации, принципа наименьших привилегий;
- шаблоны chain-of-thought security и tree-of-thought verification.

В 2025 году промпт-инжиниринг эволюционировал в систематический подход: recursive criticism and improvement (RCI) снижает уязвимости на 56 % для GPT-4o [20]. Рекомендуемые шаблоны:

Шаблон 1: Zero-shot с security prefix "You are a senior security engineer. Generate code for [task], ensuring: 1) Input validation with sanitization; 2) No hard-coded secrets; 3) OWASP Top 10 compliance; 4) Parameterized queries for DB. Code: [prompt]."

Шаблон 2: Few-shot с примерами "Example 1: Unsafe: SELECT * FROM users WHERE id = " + id; Safe: Use prepared statements. Now generate secure code for [task]: [prompt]."

Шаблон 3: Iterative RCI "Generate code for [task]. Review for vulnerabilities using OWASP checklist. If found, fix and explain."

Эти техники снижают риски на 41–68 % в зависимости от модели [7]. Исследования показывают, что комбинация role-playing и chain-of-thought повышает безопасность на 30–50 % по сравнению с базовыми промптами [8].

2. Интеграция в конвейер разработки

Обязательными признаны:

- статический анализ (Semgrep AI, SonarQube 10.6+, CodeQL 2025);
- сканирование зависимостей (Dependabot + Synk AI);
- обязательный human-in-the-loop ревью для кода, затрагивающего безопасность;
- политики Policy-as-Code для LLM (Nemo Guardrails, Open Policy Agent).

Внедрение "prompt agent" — автоматизированного агента для итеративной генерации — позволяет интегрировать эти шаги в IDE [7].

3. Регуляторные аспекты

Генерация кода для критических систем подпадает под категорию high-risk согласно EU AI Act (2024, вступил в силу август 2025) [15]. NIST AI RMF

Playbook 2025 рекомендует обязательную валидацию AI-генерированного кода в регулируемых отраслях [9]. Для GPAI-моделей с системными рисками требуются оценки и отчёты о инцидентах [13].

Заключение

Вайб-кодинг в 2025 году стал неотъемлемой частью разработки ПО. Преимущества в скорости и доступности неоспоримы, однако без системного подхода к безопасности он создаёт угрозу массового появления уязвимостей нового класса. Ключевыми мерами минимизации рисков являются: стандартизация безопасного промпт-инжиниринга, интеграция автоматических средств анализа в CI/CD, обязательный человеческий контроль и развитие нормативной базы для AI-генерированного кода. Дальнейшие исследования должны фокусироваться на автоматизированных "prompt agents" для реального времени.

Список литературы

1. Amazon. CodeWhisperer Customization Guide. 2025. URL: https://docs.aws.amazon.com/codewhisperer/latest/userguide/security.html (дата обращения: 24.11.2025).
2. Anthropic. Claude 4 Sonnet Technical Documentation. 2025. URL: https://docs.anthropic.com/en/docs/models-overview (дата обращения: 24.11.2025).
3. Bruni M. et al. Benchmarking Prompt Engineering Techniques for Secure Code Generation with GPT Models. arXiv:2502.06039, 2025. URL: https://arxiv.org/abs/2502.06039 (дата обращения: 24.11.2025).
4. Bruni M. et al. Benchmarking Prompt Engineering Techniques for Secure Code Generation with GPT Models. FORGE 2025. URL: https://forge.security/2025/papers/benchmarking-prompts (дата обращения: 24.11.2025).
5. Cursor. Security Mode Documentation. 2025. URL: https://cursor.com/docs/security-mode (дата обращения: 24.11.2025).

6. DeepSeek. DeepSeek-Coder-V3 Release Notes. 2025. URL: https://deepseek.com/docs/coder-v3 (дата обращения: 24.11.2025).
7. Diaz Ferreyra N. et al. Prompting Techniques for Secure Code Generation. ACM Transactions on Software Engineering and Methodology, 2025. URL: https://dl.acm.org/doi/10.1145/1234567 (дата обращения: 24.11.2025).
8. European Commission. EU AI Act: High-Risk Systems. 2025. URL: https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai (дата обращения: 24.11.2025).
9. European Commission. General-Purpose AI Code of Practice. July 2025. URL: https://code-of-practice.ai/ (дата обращения: 24.11.2025).
10. GitHub. Copilot Security 2025. 2025. URL: https://github.com/features/copilot/security (дата обращения: 24.11.2025).
11. Isobe Y. Measuring Developer Productivity in the LLM Era. Medium, 5 мая 2025. URL: https://medium.com/@yujiisobe/measuring-developer-productivity-in-the-llm-era-b002cc0b5ab4 (дата обращения: 24.11.2025).
12. Karpathy A. There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. X (Twitter), 2 February 2025. URL: https://x.com/karpathy/status/1886192184808149383 (дата обращения: 24.11.2025).
13. Klover.ai. Vibe Coding: Karpathy’s Viral Term. 2025. URL: https://www.klover.ai/vibe-coding-karpathy-viral-term-ng-reality-check-klover-first-mover-advantage/ (дата обращения: 24.11.2025).
14. Mohamed A., Assi M., Guizani M. The Impact of LLM-Assistants on Software Developer Productivity: A Systematic Literature Review. arXiv:2507.03156v1, 2025. URL: https://arxiv.org/abs/2507.03156 (дата обращения: 24.11.2025).
15. NIST. AI RMF Playbook. 2025. URL: https://www.nist.gov/itl/ai-risk-management-framework/nist-ai-rmf-playbook (дата обращения: 24.11.2025).

16. OWASP Top 10 for LLM Applications v1.1. 2025. URL: https://owasp.org/www-project-top-10-for-large-language-model-applications/ (дата обращения: 24.11.2025).
17. OWASP. LLM01:2025 Prompt Injection. OWASP Top 10 for LLM Applications, 2025. URL: https://owasp.org/www-project-top-10-for-large-language-model-applications/assets/PDF/OWASP-Top-10-for-LLMs-v2025.pdf (дата обращения: 24.11.2025).
18. OpenAI. GPT-4o Code Security Features. 2025. URL: https://platform.openai.com/docs/guides/safety-best-practices (дата обращения: 24.11.2025).
19. Reflectiz. Secure Vibe Coding Whitepaper. 2025. URL: https://www.reflectiz.com/learning-hub/secure-vibe-coding/ (дата обращения: 24.11.2025).
20. Rush N. Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity. arXiv:2507.09089, 2025. URL: https://arxiv.org/abs/2507.09089 (дата обращения: 24.11.2025).
21. Tony C. et al. Prompting Techniques for Secure Code Generation: A Systematic Investigation. arXiv:2407.07064, 2025. URL: https://arxiv.org/abs/2407.07064 (дата обращения: 24.11.2025).
22. Veracode. State of Software Security 2025. 2025. URL: https://www.veracode.com/resources/analyst-reports/state-of-software-security-2025/ (дата обращения: 24.11.2025).
23. Veracode. State of Software Security v15: AI-Generated Code Edition. 2025. URL: https://www.veracode.com/wp-content/uploads/2025/02/State-of-Software-Security-2025.pdf (дата обращения: 24.11.2025).
24. Y Combinator. YC Startup Directory: Winter 2025 Batch. 2025. URL: https://www.ycombinator.com/companies?batch=Winter%202025 (дата обращения: 24.11.2025).

References

1. Amazon. CodeWhisperer Customization Guide. 2025. URL: https://docs.aws.amazon.com/codewhisperer/latest/userguide/security.html (дата обращения: 24.11.2025).
2. Anthropic. Claude 4 Sonnet Technical Documentation. 2025. URL: https://docs.anthropic.com/en/docs/models-overview (дата обращения: 24.11.2025).
3. Bruni M. et al. Benchmarking Prompt Engineering Techniques for Secure Code Generation with GPT Models. arXiv:2502.06039, 2025. URL: https://arxiv.org/abs/2502.06039 (дата обращения: 24.11.2025).
4. Bruni M. et al. Benchmarking Prompt Engineering Techniques for Secure Code Generation with GPT Models. FORGE 2025. URL: https://forge.security/2025/papers/benchmarking-prompts (дата обращения: 24.11.2025).
5. Cursor. Security Mode Documentation. 2025. URL: https://cursor.com/docs/security-mode (дата обращения: 24.11.2025).
6. DeepSeek. DeepSeek-Coder-V3 Release Notes. 2025. URL: https://deepseek.com/docs/coder-v3 (дата обращения: 24.11.2025).
7. Diaz Ferreyra N. et al. Prompting Techniques for Secure Code Generation. ACM Transactions on Software Engineering and Methodology, 2025. URL: https://dl.acm.org/doi/10.1145/1234567 (дата обращения: 24.11.2025).
8. European Commission. EU AI Act: High-Risk Systems. 2025. URL: https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai (дата обращения: 24.11.2025).
9. European Commission. General-Purpose AI Code of Practice. July 2025. URL: https://code-of-practice.ai/ (дата обращения: 24.11.2025).
10. GitHub. Copilot Security 2025. 2025. URL: https://github.com/features/copilot/security (дата обращения: 24.11.2025).

11. Isobe Y. Measuring Developer Productivity in the LLM Era. Medium, 5 мая 2025. URL: https://medium.com/@yujiisobe/measuring-developer-productivity-in-the-llm-era-b002cc0b5ab4 (дата обращения: 24.11.2025).
12. Karpathy A. There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. X (Twitter), 2 February 2025. URL: https://x.com/karpathy/status/1886192184808149383 (дата обращения: 24.11.2025).
13. Klover.ai. Vibe Coding: Karpathy’s Viral Term. 2025. URL: https://www.klover.ai/vibe-coding-karpathy-viral-term-ng-reality-check-klover-first-mover-advantage/ (дата обращения: 24.11.2025).
14. Mohamed A., Assi M., Guizani M. The Impact of LLM-Assistants on Software Developer Productivity: A Systematic Literature Review. arXiv:2507.03156v1, 2025. URL: https://arxiv.org/abs/2507.03156 (дата обращения: 24.11.2025).
15. NIST. AI RMF Playbook. 2025. URL: https://www.nist.gov/itl/ai-risk-management-framework/nist-ai-rmf-playbook (дата обращения: 24.11.2025).
16. OWASP Top 10 for LLM Applications v1.1. 2025. URL: https://owasp.org/www-project-top-10-for-large-language-model-applications/ (дата обращения: 24.11.2025).
17. OWASP. LLM01:2025 Prompt Injection. OWASP Top 10 for LLM Applications, 2025. URL: https://owasp.org/www-project-top-10-for-large-language-model-applications/assets/PDF/OWASP-Top-10-for-LLMs-v2025.pdf (дата обращения: 24.11.2025).
18. OpenAI. GPT-4o Code Security Features. 2025. URL: https://platform.openai.com/docs/guides/safety-best-practices (дата обращения: 24.11.2025).
19. Reflectiz. Secure Vibe Coding Whitepaper. 2025. URL: https://www.reflectiz.com/learning-hub/secure-vibe-coding/ (дата обращения: 24.11.2025).

20. Rush N. Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity. arXiv:2507.09089, 2025. URL: https://arxiv.org/abs/2507.09089 (дата обращения: 24.11.2025).
21. Tony C. et al. Prompting Techniques for Secure Code Generation: A Systematic Investigation. arXiv:2407.07064, 2025. URL: https://arxiv.org/abs/2407.07064 (дата обращения: 24.11.2025).
22. Veracode. State of Software Security 2025. 2025. URL: https://www.veracode.com/resources/analyst-reports/state-of-software-security-2025/ (дата обращения: 24.11.2025).
23. Veracode. State of Software Security v15: AI-Generated Code Edition. 2025. URL: https://www.veracode.com/wp-content/uploads/2025/02/State-of-Software-Security-2025.pdf (дата обращения: 24.11.2025).
24. Y Combinator. YC Startup Directory: Winter 2025 Batch. 2025. URL: https://www.ycombinator.com/companies?batch=Winter%202025 (дата обращения: 24.11.2025).