



АВТОМАТИЗАЦИЯ ОЦЕНКИ ВРЕДНОСТИ СКРИПТОВ НА ОСНОВЕ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ ПРИ РАССЛЕДОВАНИИ КОМПЬЮТЕРНЫХ ИНЦИДЕНТОВ

В статье рассматривается возможность автоматизации одного из самых трудоёмких этапов расследования компьютерных инцидентов — анализа скриптов на предмет вредности. Для этого нами разработан специализированный датасет, включающий 98 скриптов на языках Python, PowerShell и Bash, размеченных экспертами на «вредные» и «безвредные». На основе этого датасета проведено сравнительное исследование проприетарной модели gpt-4-turbo и открытой LLaMA-3.1-70B с использованием подхода Chain-of-Thought (CoT) для повышения интерпретируемости выводов. Эксперимент показал, что LLaMA-3.1-70B достигла 100% точности ($F1=1,0$), хотя такой результат, вероятно, связан с ограниченным составом датасета. Модель gpt-4-turbo продемонстрировала более умеренные, но реалистичные метрики ($F1=0,72$), однако с высоким числом ложноположительных срабатываний. Полученные данные подтверждают перспективность интеграции больших языковых моделей в процессы цифровой криминалистики, при этом подчёркивают критическую важность учёта контекста выполнения скриптов.

Ключевые слова: защита информации, кибербезопасность, большие языковые модели, анализ скриптов, расследование инцидентов, датасет, машинное обучение в кибербезопасности.

AUTOMATING THE ASSESSMENT OF SCRIPT MALICIOUSNESS BASED ON LARGE LANGUAGE MODELS WHEN INVESTIGATING COMPUTER INCIDENTS

This paper explores the automation of one of the most labor-intensive stages of computer incident investigation—the analysis of script maliciousness. To this end, we developed a specialized dataset comprising 98 expert-labeled scripts in Python, PowerShell, and Bash, categorized as 'malicious' or 'benign'. Using this dataset, a comparative study was conducted between the proprietary gpt-4-turbo model and the open-source LLaMA-3.1-70B, employing a Chain-of-Thought (CoT) approach to enhance the interpretability of the conclusions. The experiment showed that LLaMA-3.1-70B achieved 100% accuracy (F1=1.0), a result likely attributed to the limited composition of the dataset. The gpt-4-turbo model demonstrated more moderate yet realistic metrics (F1=0.72), but with a high number of false positives. The findings confirm the promise of integrating Large Language Models into digital forensics processes while highlighting the critical importance of considering script execution context.

Keywords: information security, cybersecurity, large language models (LLMs), script analysis, incident response, dataset, machine learning.

Введение. При расследовании компьютерных инцидентов особое внимание уделяется цифровым следам, оставленным злоумышленниками. Среди них особенно ценными оказываются скрипты — последовательности команд на языках Python, PowerShell или Bash, которые атакующие используют для автоматизации своих действий: от первоначального проникновения до перемещения по инфраструктуре и выполнения вредоносных операций. Анализ таких скриптов позволяет не только выявить факт компрометации, но и восстановить тактики, техники и процедуры (TTPs), применённые злоумышленниками, а также оценить потенциальный ущерб. Однако в условиях роста числа и сложности инцидентов ручной анализ становится всё менее жизнеспособным. Это объективно подталкивает к поиску решений на основе автоматизации — и здесь большие языковые модели (Large Language Models, LLM) выглядят многообещающе. Они способны не только классифицировать скрипты по степени вредоносности, но и выделять индикаторы ком-

прометации (IoC) и формировать аналитические заключения для специалистов по кибербезопасности.

Эффективность применения LLM напрямую зависит от качества обучающих и тестовых данных. В нашей работе мы предложили оригинальный подход к формированию датасета, включающего как скрипты из реальных инцидентов, так и легитимные примеры, используемые в системном администрировании, с последующей экспертной разметкой.

Важно отметить, что использование проприетарных моделей, таких как gpt-4-turbo или Claude 3, несмотря на их высокую точность, сопряжено с рисками утечки конфиденциальной информации, поскольку запросы обрабатываются на стороне облачного провайдера. В задачах информационной безопасности предпочтение целесообразно отдавать открытым LLM — таким как LLaMA [1], Mistral [2] и Qwen [3], — которые могут быть развернуты в изолированных средах и поддерживают fine-tuning под специфические сценарии [4].

На сегодняшний день применение LLM в кибербезопасности активно изучается, но результаты остаются неоднозначными. Например, GPT-4 показывает высокую эффективность в исправлении уязвимого кода, снижая их количество на 90% [5]. В работе [6] отмечается, что LLM успешно идентифицируют такие уязвимости, как SQL-инъекции и XSS, и дают рекомендации по их устранению. Интеграция LLM с формальными методами верификации позволяет устранить до 80% уязвимостей [7], хотя такой подход требует значительных вычислительных ресурсов. Вместе с тем существуют и ограничения. Исследование [8] показало низкую точность LLM при обнаружении сложных уязвимостей типа zero-day из-за недостатка контекста выполнения. Кроме того, даже при превосходстве GPT-4 над специализированными моделями, такими как CodeBERT [9], её использование ограничено в сценариях, где критична конфиденциальность.

Нами выделены три ключевых пробела в современных исследованиях. Во-первых, сохраняется дефицит репрезентативных датасетов, сочетающих вредоносные и легитимные скрипты из реальных инцидентов. Во-вторых, недостаточно изучена эффективность открытых LLM, таких как LLaMA, в задачах автоматизации расследований. В-третьих, практически не исследовано влияние контекста выполнения на достоверность классификации скриптов.

Целью настоящей работы является экспериментальная оценка достоверности больших языковых моделей в анализе скриптов и обоснование целесообразности их применения для автоматизации этапов расследования компьютерных инцидентов. В отличие от предыдущих работ, мы сосредоточились именно на скриптах (а не на произвольном коде), сравнили проприетарные и открытые модели, а также проанализировали влияние контекста выполнения — фактор, критически важный при анализе реальных инцидентов.

Описание датасета. Для проведения экспериментов нами был создан специализированный датасет, включающий 98 скриптов на языках Python, PowerShell и Bash. Каждый скрипт сохранён в виде отдельного текстового файла и отражает реальные сценарии, с которыми сталкиваются специалисты при расследовании инцидентов. При формировании датасета мы руководствовались практическими потребностями анализа: в него вошли как легитимные скрипты, используемые в повседневной работе системных администраторов и DevOps-инженеров, так и примеры, характерные для действий злоумышленников. Все скрипты были размечены экспертами и распределены по двум категориям:

«Безвредные» — легитимные сценарии (например, резервное копирование, развёртывание приложений, мониторинг ресурсов);

«Вредные» — команды, направленные на компрометацию систем (сбор учётных данных, обход защиты, эксфильтрация информации).

По функциональному происхождению скрипты делятся на три группы:

DevOps-скрипты [10] — автоматизация CI/CD-процессов, управление контейнерами, развёртывание инфраструктуры;

Скрипты системного администрирования — настройка ОС, управление пользователями, диагностика;

Скрипты, используемые в атаках — примеры из реальных инцидентов, включая техники, описанные в MITRE ATT&CK.

Итоговое распределение представлено в Таблице 1.

Важно отметить одно ограничение: в текущей версии датасета отсутствует контекст выполнения скриптов (окружение, права пользователя, сетевые условия). Это сознательное упрощение, моделирующее начальный этап расследования, когда аналитик имеет доступ только к самому коду.

Пример вредоносного скрипта на рисунке 1 демонстрирует типичную цепочку дей-

Таблица 1

Распределение скриптов по категориям

№ п/п	Категория	Количество скриптов	Процент от общего числа
1	Безвредный	58	59%
2	Вредный	40	41%

```

1 @echo off
2 mode con: cols=50 lines=30
3 cls
4 setlocal
5 setlocal enabledelayedexpansion
6 cd /d %~dp0
7 mkdir c:\users
8 bcdedit /deletevalue {current} safeboot
9 bcdedit /deletevalue safeboot
10 c:\users\7z.exe x c:\users\C.7z -pBH$ghN12 -y -oc:\users
11 del /f /q c:\users\C.7z
12 Reg DELETE "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v AutoAdminLogon /f
13 Reg DELETE "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v ForceAutoLogon /f
14 Reg DELETE "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultUserName /f
15 Reg DELETE "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v DefaultPassword /f
16 Reg DELETE "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell /f
17 Reg DELETE "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v ShellMaintenances /f
18 Reg DELETE HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce /v *ShellMaintenance /f
19 Reg DELETE HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v *ShellMaintenac /f
20 start c:\users\svhost.exe -mode costum -priority off -power restart
21 del /f /q c:\users\fin3.bat
22 DEL %0 /q .exe /F
23 exit

```

Рис.1. Листинг вредоносного скрипта

ствий: взаимодействие с процессом LSASS, создание дампа памяти и его архивация для последующей передачи за пределы системы.

Методика анализа скриптов. Для повышения надёжности и интерпретируемости анализа мы применили метод Chain-of-Thought (CoT) [11–13]. В отличие от прямой классификации, CoT заставляет модель «думать вслух»: сначала она подробно объясняет, что делает каждая команда в скрипте, а затем на основе этого анализа формирует вывод о его вредоносности. Такой двухэтапный подход позволяет не только получить ответ, но и проследить логику рассуждений модели — что критически важно при расследовании инцидентов.

Для управления поведением модели был разработан системный промт [14–15], который явно указывает, на какие признаки следует обращать внимание. В частности, промт направляет модель на выявление следующих характеристик:

- попыток несанкционированного доступа к критическим системным файлам или процессам (например, LSASS);

- инициации сетевых соединений с подозрительными или внешними адресами;

- использования команд повышения привилегий или обхода механизмов защиты;

- действий, направленных на извлечение, шифрование или передачу конфиденциальных данных.

Системный промт

Вы являетесь экспертом в области кибербезопасности.

Ваша задача — проанализировать предоставленный скрипт и определить, является ли он вредоносным или безвредным.

Пожалуйста, следуйте этим шагам:

1. Прочитайте содержимое скрипта, который будет предоставлен ниже.
2. Объясните ваши шаги и логику, которые вы используете для определения вредоносности скрипта.
3. Укажите признаки вредоносного поведения, такие как:
 - Попытки доступа или модификации чувствительных системных файлов.
 - Попытки подключения к внешним серверам или загрузки вредоносных полезных нагрузок.
 - Попытки повышения привилегий или обхода средств безопасности.
 - Эксфильтрация данных или шифрование данных.
4. На основе вашего анализа дайте окончательный вывод о характеристиках скрипта

Рис.2. Пример системного промта



Рис.3. Общая схема метода анализа скриптов

Пример такого промта приведён на рисунке 2, а общая схема процесса — на рисунке 3.

Ниже — реальный вывод модели при анализе вредоносного PowerShell-скрипта:

Основываясь на анализе команды PowerShell, я делаю вывод, что выполняемые действия, вероятно, являются злонамеренными.

Вот почему:

1. Несанкционированный доступ к системе:

Команда взаимодействует с системными файлами и процессами, такими как подсистема безопасности Local Security Authority Subsystem Service (LSASS), которая является критическим системным процессом. Это предполагает, что команда пытается получить доступ к чувствительным системным ресурсам.

2. Сбор дампа памяти:

Команда создаёт полный дамп памяти процесса LSASS, который может содержать чувствительную информацию, такую как пароли, ключи шифрования или другие конфиденциальные данные. Это может быть использовано для злонамеренных целей, таких как извлечение чувствительной информации или эксплуатация уязвимостей.

3. Эксфильтрация данных:

Команда сжимает дамп памяти в архив ZIP, который может быть использован для эксфильтрации данных из системы. Это явля-

ется распространённой техникой, используемой атакующими для кражи чувствительной информации.

4. Уклончивое поведение:

Использование параметров `-ErrorAction Ignore` и `-Force` предполагает, что команда спроектирована для работы скрытно и избегания обнаружения. Это является общей характеристикой злонамеренного кода. Хотя цель команды может быть законной, например, отладкой или устранением неисправностей, способ её конструкции и выполняемые действия предполагают, что она может быть использована для злонамеренных целей. Поэтому я делаю вывод, что действия, выполняемые этой командой, вероятно, являются злонамеренными.

Этот пример наглядно демонстрирует, как модель корректно идентифицирует опасные операции с процессом LSASS и последующую эксфильтрацию данных — даже без доступа к контексту выполнения.

Результаты. Мы провели сравнительную оценку двух больших языковых моделей — gpt-4-turbo (версия от 2024-04-09) и LLaMA-3.1-70B — в задаче бинарной классификации скриптов. Выбор gpt-4-turbo обусловлен её признанным лидерством в reasoning [16–17] и code review [18–20], а LLaMA-3.1-70B — её открытой архитектурой, что критично для задач информационной безопасности. Результаты представлены в таблице 2.

Таблица 2

Результаты

№ п/п	Модель	Точность	F1-score	FP	FN
1	gpt-4-turbo	80.6%	0.72	16	2
2	LLaMA-3.1-70B	100%	1	0	0

Модель gpt-4-turbo показала заметный дисбаланс в распознавании клас-сов: из 58 безвредных скриптов она, верно, определила 56 (96,5%), но из 40 вредоносных — лишь 24 (60%). При этом 16 ложноположительных срабатываний (FP) оказались связаны в основном с агрессивными, но легитимными DevOps-командами, например, kill-9. Это подтверждает гипотезу: без учёта контекста выполнения даже передовые модели склонны интерпретировать стандартные административные действия как угрозу.

В то же время LLaMA-3.1-70B продемонстрировала идеальные метрики: все 98 скриптов были классифицированы верно. Хотя такой результат выглядит впечатляюще, мы считаем его потенциально завышенным. В реальных условиях редко удаётся достичь 100% точности в задачах классификации, особенно при ограниченном размере датасета и отсутствии обфусцированного кода. Вероятно, модель «запомнила» паттерны из обучающей выборки или столкнулась с недостаточной вариативностью вредоносных примеров. Тем не менее, даже с учётом этих оговорок, эксперимент подтверждает: открытые LLM способны конкурировать с проприетарными аналогами в узкоспециализированных задачах, при этом обеспечивая полный контроль над данными.

Заключение. Наши эксперименты показывают, что большие языковые модели дей-

ствительно могут стать полезным инструментом при автоматизации анализа скриптов в ходе расследования компьютерных инцидентов. Модель LLaMA-3.1-70B продемонстрировала идеальную точность (100%, F1=1,0), однако мы считаем этот результат условным — он, скорее всего, объясняется ограниченным составом датасета: небольшой объём (98 скриптов) и отсутствие обфусцированного кода не позволяют делать выводы о реальной применимости в сложных сценариях. Для валидации требуются тесты на расширенной и более разнообразной выборке. В то же время gpt-4-turbo, несмотря на более скромные метрики (F1=0,72), показала более сбалансированное и реалистичное поведение. Высокая доля ложноположительных срабатываний (FP=16) в основном связана с агрессивными, но легитимными DevOps-командами. Это ярко иллюстрирует главный вывод нашей работы: без учёта контекста выполнения скрипта даже самые передовые модели будут ошибаться.

Перспективные направления дальнейших исследований включают разработку датасетов с обфусцированным кодом, интеграцию LLM с системами сбора контекста и создание гибридных методов анализа. Практическое внедрение может быть в разработке плагинов для SIEM-систем на основе LLM, в автоматизации генерации отчетов с рекомендациями по реагированию.

Литература

1. Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971v1, 2023.
2. Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Léo Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, William El Sayed. Mixtral of Experts. arXiv:2401.04088v1, 2024.
3. Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, Tianhang Zhu. QWEN TECHNICAL REPORT. arXiv:2309.16609v1, 2023.
4. Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Khan, and Arsalan Shahid. The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities. arXiv:2408.13296v3, 2024.
5. David Noever. Can large language models find and fix vulnerable soft-ware? arXiv preprint arXiv:2308.10345, 2023.

6. Haonan Li, Yu Hao, Yizhuo Zhai, and Zhiyun Qian. Assisting static analysis with large language models: A chatgpt experiment. In Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, pages 2107–2111, New York, NY, USA, 2023.
7. Yiannis Charalambous, Norbert Tihanyi, Ridhi Jain, Youcheng Sun, Mohamed Amine Ferrag, and Lucas C Cordeiro. A new era in software security: Towards self-healing software via large language models and formal verification. arXiv preprint arXiv:2305.14752, 2023.
8. Zibin Zheng, Kaiwen Ning, Jiachi Chen, Yanlin Wang, Wenqing Chen, Lianghong Guo, and Weicheng Wang. Towards an understanding of large language models in software engineering tasks. arXiv preprint arXiv:2308.11396, 2023.
9. Xin Zhou, Ting Zhang, and David Lo. Large language model for vulnerability detection: emerging results and future directions. arXiv preprint arXiv:2401.15468, 2024.
10. Mayank Gokarna, Raju Singh. DevOps: A Historical Review and Future Works. 2012.06145, 2012
11. Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903v6, 2023.
12. Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, Huan Sun. Towards Understanding Chain-of-Thought Prompting: An Empirical Study of What Matters. arXiv:2212.10001v2, 2023.
13. Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, Yusuke Iwasawa. Large Language Models are Zero-Shot Reasoners. arXiv:2205.11916v4, 2023.
14. Kaiyan Chang, Songcheng Xu, Chenglong Wang, Yingfeng Luo, Tong Xiao, Jingbo Zhu. Efficient Prompting Methods for Large Language Models: A Survey. arXiv:2404.01077v1, 2024.
15. Shubham Vatsal, Harsh Dubey. A Survey of Prompt Engineering Methods In Large Language Models For Different NLP Tasks. arXiv:2407.12994, 2024.
16. Aske Laat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, Thomas Back. Reasoning with Large Language Models, a Survey. arXiv:2407.11511v1, 2024.
17. Shibo Hao, Yi Gu, Haotian Luo, Tianyang Liu, Xiyan Shao, Xinyuan Wang, Shuhua Xie, Haodi Ma, Adithya Samavedhi, Qiyue Gao, Zhen Wang, Zhiting Hu. New Evaluation, Library, and Analysis of Step-by-Step Reasoning with Large Language Models. arXiv:2404.05221v2, 2024.
18. Jiaxin Yu, Peng Liang, Yujia Fu, Amjed Tahir, Mojtaba Shahin, Chong Wang, Yangxiao Cai. An Insight into Security Code Review with LLMs: Capabilities, Obstacles and Influential Factors. arXiv:2401.16310v3, 2024.
19. Zeeshan Rasheed, Malik Abdul Sami, Muhammad Waseem, Kai-Kristian Kemell, Xiaofeng Wang, Anh Nguyen, Kari Systä and Pekka Abrahamsson. AI-powered Code Review with LLMs: Early Results. arXiv:2404.18496v1, 2024.
20. Jiaxin Yu, Peng Liang, Yujia Fu, Mojtaba Shahin, Amjed Tahir, Chong Wang, Yangxiao Cai. Security Code Review by Large Language Models. arXiv:2401.16310v2, 2024.

References

1. Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971v1, 2023.
2. Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Léo Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, William El Sayed. Mixtral of Experts. arXiv:2401.04088v1, 2024.
3. Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, Tianhang Zhu. QWEN TECHNICAL REPORT. arXiv:2309.16609v1, 2023.
4. Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Khan, and Arsalan Shahid. The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities. arXiv:2408.13296v3, 2024.

5. David Noever. Can large language models find and fix vulnerable soft-ware? arXiv preprint arXiv:2308.10345, 2023.
6. Haonan Li, Yu Hao, Yizhuo Zhai, and Zhiyun Qian. Assisting static analysis with large language models: A chatgpt experiment. In Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, pages 2107–2111, New York, NY, USA, 2023.
7. Yiannis Charalambous, Norbert Tihanyi, Ridhi Jain, Youcheng Sun, Mohamed Amine Ferrag, and Lucas C Cordeiro. A new era in software security: Towards self-healing software via large language models and formal verification. arXiv preprint arXiv:2305.14752, 2023.
8. Zibin Zheng, Kaiwen Ning, Jiachi Chen, Yanlin Wang, Wenqing Chen, Lianghong Guo, and Weicheng Wang. Towards an understanding of large language models in software engineering tasks. arXiv preprint arXiv:2308.11396, 2023.
9. Xin Zhou, Ting Zhang, and David Lo. Large language model for vulne-rability detection: emerging results and future directions. arXiv preprint ar-Xiv:2401.15468, 2024.
10. Mayank Gokarna, Raju Singh. DevOps: A Historical Review and Fu-ture Works. 2012.06145, 2012
11. Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv: 2201.11903v6, 2023.
12. Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, Huan Sun. Towards Understanding Chain-of-Thought Prompting: An Empirical Study of What Matters. arXiv:2212.10001v2, 2023.
13. Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, Yusuke Iwasawa. Large Language Models are Zero-Shot Reasoners. ar-Xiv:2205.11916v4, 2023.
14. Kaiyan Chang, Songcheng Xu, Chenglong Wang, Yingfeng Luo, Tong Xiao, Jingbo Zhu. Efficient Prompting Methods for Large Language Models: A Survey. arXiv:2404.01077v1, 2024.
15. Shubham Vatsal, Harsh Dubey. A Survey of Prompt Engineering Me-thods In Large Language Models For Different Nlp Tasks. arXiv:2407.12994, 2024.
16. Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, Thomas Back. Reasoning with Large Language Models, a Survey. ar-Xiv:2407.11511v1, 2024.
17. Shibo Hao, Yi Gu, Haotian Luo, Tianyang Liu, Xiyang Shao, Xinyuan Wang, Shuhua Xie, Haodi Ma, Adithya Samavedhi, Qiyue Gao, Zhen Wang, Zhiting Hu. New Evaluation, Library, and Analysis of Step-by-Step Reasoning with Large Language Models. arXiv:2404.05221v2, 2024.
18. Jiaxin Yu, Peng Liang, Yujia Fu, Amjed Tahir, Mojtaba Shahin, Chong Wang, Yangxiao Cai. An Insight into Security Code Review with LLMs: Capabilities, Obstacles and Influential Factors. arXiv:2401.16310v3, 2024.
19. Zeeshan Rasheedl, Malik Abdul Sami, Muhammad Waseem, Kai-Kristian Kemell, Xiaofeng Wang, Anh Nguyen, Kari Systä and Pekka Abrahamsson. AI-powered Code Review with LLMs: Early Results. arXiv:2404.18496v1, 2024.
20. Jiaxin Yu, Peng Liang, Yujia Fu, Mojtaba Shahin, Amjed Tahir, Chong Wang, Yangxiao Cai. Security Code Review by Large Language Models. arXiv:2401.16310v2, 2024.

МЕЛЬНИКОВ Андрей Витальевич, доктор технических наук, профессор, директор, автономное учреждение Ханты-Мансийского автономного округа – Югры «Югорский научно-исследовательский институт информационных технологий». 628011, г. Ханты-Мансийск, ул. Мира, 151. E-mail: melnikovav@uriit.ru

ЯРЫШКИН Иван Николаевич, начальник Управления кибербезопасности, автономное учреждение Ханты-Мансийского автономного округа – Югры «Югорский научно-исследовательский институт информационных технологий». 628011, г. Ханты-Мансийск, ул. Мира, 151. E-mail: yin@uriit.ru

MELNIKOV Andrey Vitalievich, Doctor of Technical Sciences, Professor, Director, Ugra Research Institute of Information Technologies. 628011, Khanty-Mansiysk, Mira str., 151. Email: melnikovav@uriit.ru

YARYSHKIN Ivan Nikolaevich, Head of the Cybersecurity Department, Ugra Research Institute of Information Technologies. 628011, Khanty-Mansiysk, Mira str., 151. Email: yin@uriit.ru