СТАТЬЯ

УДК 004.89:004.43
DOI 10.17513/snt.40140

АНАЛИЗ ЭФФЕКТИВНОСТИ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ ДЛЯ ОБНАРУЖЕНИЯ ЖЕСТКО ЗАКОДИРОВАННЫХ ПАРОЛЕЙ В ПРОГРАММНОМ КОДЕ

Швыров В.В., Капустин Д.А., Сентяй Р.Н., Шулика Т.И.

ФГБОУ ВО «Луганский государственный педагогический университет», Луганск, e-mail: slshj@yandex.ru

Статья посвящена анализу эффективности больших языковых моделей для обнаружения жестко закодированных паролей в программном коде. Установлено, что использование жестко вписанных паролей или криптографических ключей в программном коде может привести к серьезным проблемам безопасности и потенциальным утечкам данных. Наиболее распространенным способом выявления таких проблем является метод статического анализа программного кода. Одним из перспективных направлений представляется использование различных больших языковых моделей для анализа кода и поиска уязвимостей. В то же время полное переобучение моделей сопряжено со значительными затратами ресурсов. В связи с этим развитие получили различные альтернативные техники обучения, которые позволяют адаптировать большие языковые модели путем тонкой настройки части параметров. В работе для обучения моделей использована техника низкоранговой адаптации (LoRa), а также авторский набор данных для обучения выявлению жестко вписанных паролей в программном коде. Кроме того, выполнена оценка адекватности обученных моделей, определены количественные показатели ложно положительных, ложно отрицательных прогнозов модели на тестовом наборе данных, а также вычислены значения полноты, точности и F-меры и определены большие языковые модели, наиболее эффективные для детектирования жестко вписанных паролей при тонкой настройке с использованием техники LoRA.

Ключевые слова: большие языковые модели, жестко вписанные пароли, программный код, статический анализ, LoRA, Python

Работа выполнена в рамках государственного задания Министерства просвещения Российской Федерации для ФГБОУ ВО «Луганский государственный педагогический университет».

EVALUATION OF THE EFFECTIVENESS OF LARGE LANGUAGE MODELS FOR DETECTING HARD-CODED PASSWORDS IN SOURCE CODE

Shvyrov V.V., Kapustin D.A., Sentyay R.N., Shulika T.I.

Lugansk State Pedagogical University, Lugansk, e-mail: slshj@yandex.ru

The article is devoted to the analysis of the effectiveness of large language models for detecting hard-coded passwords in program code. It has been established that the use of hard-coded passwords or cryptographic keys in software code can lead to serious security issues and potential data leaks. The most common method of identifying such problems is the method of static analysis of the program code. One of the promising directions is the use of various large language models for code analysis and the search for vulnerabilities. At the same time, complete retraining of models is associated with significant resource costs. In this regard, various alternative learning techniques have been developed that allow adapting large language models by fine-tuning some of the parameters. In the work, the technique of low-rank adaptation (LoRa) was used to train the models, as well as the author's data set for training the identification of hard-coded passwords in the program code. In addition, an assessment of the adequacy of the trained models was performed, the quantitative indicators of false positive and false negative predictions of the model on the test data set were determined, as well as the values of completeness, accuracy and F-measure were calculated, and large language models were determined to be the most effective for detecting hard-coded passwords with thin setup using LoRA technology.

Keywords: large language models, hardcoded passwords, program code, static analysis, LoRA, Python

The work was carried out within the framework of the state assignment of the Ministry of Education of the Russian Federation for the Lugansk State Pedagogical University.

Введение

По данным рейтинга OWASP TOP 10 [1] проблемы, связанные со сбоями аутентификации и авторизации пользователей, занимают 7 место (A07:2021 – Identification and Authentication Failures). Список различных способов реализации угроз и типов уязвимостей по каталогу CWE [2], которые могут быть сопоставлены данным проблемам, достаточно обширный. В частности, можно выделить группу уязвимостей, которые связаны с использованием жестко закодированных учетных данных пользователей CWE-798, CWE-259. Следуя отечественному каталогу БДУ ФСТЭК и новому разделу угроз [3], данная группа способов реализации угроз представлена как СП.17 – «Подбор (восстановление) аутентификационной информации» и способ СП.17.9 «Восстановле-

ние аутентификационной информации из исходного кода/конфигурационных файлов».

Одним из распространенных методов анализа программного кода является метод статического анализа [4]. Существуют как универсальные статические анализаторы, так и специализированные для языка Python [5]. К недостаткам существующих методов для языка Python можно отнести недостаточный уровень эвристики, вследствие чего ряд потенциальных уязвимостей может быть пропущен. Таким образом, задача повышения эффективности методов поиска уязвимостей и, в частности, определения уязвимостей, связанных с жестко закодированными учетными данными в программном коде на различных императивных языках программирования, весьма актуальна.

Появление архитектуры Transformer [6] и стремительное развитие больших языковых моделей за последние годы открывает новые возможности их использования в задачах анализа программного кода. Однако полное переобучение моделей сопряжено со значительными затратами ресурсов и высокими требованиями к оборудованию. В связи с этим получили распространение различные подходы для снижения вычислительных затрат (Parameter-Efficient Fine-Tuning PEFT). Одним из таких подходов является использование техники низкоранговой адаптации моделей (Low Rank Adaptation – LoRA [7]).

Цель исследования – изучение эффективности различных нейросетевых моделей, обученных с использованием техники низкоранговой адаптации LoRA при решении задачи детектирования жестко вписанных в программный код паролей или учетных данных.

Исследование должно дать ответ на следующие вопросы:

В1. Какие модели, основанные на архитектуре Transformer, более эффективны при тонкой настройке с использованием техники низкоранговой адаптации LoRA для детектирования жестко вписанных паролей для программного кода на Python?

В2. Возможно ли использование моделей обученных для детектирования жестко вписанных паролей на Python для детектирования подобных уязвимостей на других императивных языках программирования?

На ресурсе Hugging Face [8] представлены уже более 500 тыс. различных нейросетевых моделей, в частности около 50 тыс. моделей относятся к задаче классификации текстовых данных. Количество моделей на данном ресурсе, обученных с использованием модели RoBERTa, превышает 16 тыс. Существенный прогресс в использовании различных больших языковых моделей в различных областях информационных технологий и в области статического анализа программного кода, а также значительный рост количества таких моделей делает актуальными задачи поиска эффективных методов тонкой настройки моделей в условиях дефицита ресурсов и выбора оптимальной модели для дальнейшего обучения.

Материалы и методы исследования

В работе с использованием эмпирических и статистических методов изучается возможность детектирования жестко вписанных паролей и учетных данных в программном коде, а также определяется, какая из изучаемых моделей более эффективна при ее тонкой настройке с использованием техники LoRA.

Таблица 1

Характеристики моделей

Модель	Количество параметров	Краткое описание модели
RoBERTa-base	125 313 028	RoBERTa – модель на архитектуре трансформер, предварительно обученная на большом массиве англоязычных данных
RoBERTa-large	356 610 052	Многоязычная версия RoBERTa, при обучении использовались около 2.5 TB текстов на 100 языках
CodeBERT-base	125 313 028	Получена обучением RoBERTa-base на бимодальных данных (документы и код) для задач MLM (случайное маскирование токенов) и RTD (случайное удаление токенов)
CodeBERT-base-mlm	125 313 028	Получена обучением RoBERTa-base на корпусе кода CodeSearchNet для задачи MLM
RoBERTa-MLM-based PyTorrent	90 225 412	Модель обучена на наборе данных PyTorrent в качестве базовой модели выбрана DistilBERT-Masked Language Modeling (MLM) [13]
CodeBERT-python	125 313 028	Получена обучением модели codebert-base-mlm на наборе данных codeparrot/github-code-clean для языка Python

В частности, исследуется ряд моделей, основанных на архитектуре Transformer. Проводится сравнительный анализ показателей различных моделей RoBERTa [9], CodeBERT [10], RoBERTa-large [11], CodeBERT-python [12]. В табл. 1 представлен список исследуемых моделей и их характеристики.

RoBERTa представляет собой улучшенную версию BERT с однонаправленным обучением. Она использует архитектуру Transformer, состоящую из блоков энкодеров, включающих слои внимания и полносвязные слои. Возможность тонкой настройки позволяет применять RoBERTa к различным прикладным задачам.

С использованием классических средств статического анализа авторами был сформирован набор данных, включающий строки кода на Python, которые содержат жестко вписанные пароли или учетные данные, а также набор строк, в которых данные уязвимости не были обнаружены [14]. Набор для обучения моделей был разбит на два класса – класс с меткой 0, который состоит из строк кода, не содержащих уязвимостей детектируемого вида, и класс с меткой 1, который состоит из строк кода, которые содержат уязвимости типа СП.17 (CWE-798, CWE-259). В процессе фильтрации в наборах данных были исключены пустые строки, строки комментариев, а также повторяющиеся строки. Для исключения дубликатов общий список фрагментов кода был преобразован в множество и затем сохранен в требуемом формате.

Также набор был разбит на тренировочную, тестовую и проверочную выборки в соотношении 80, 10, 10 % соответственно. Проверочная выборка не была задействована в обучении и служит для оценки адекватности и эффективности моделей. Разбиение на выборки было жестко зафиксировано в процессе разметки набора данных, что дало возможность использовать абсолютно идентичные выборки для обучения и тестирования различных моделей.

Результаты исследования и их обсуждение

Адаптация модели под конкретные задачи часто включает в себя полное переобучение, но в случае с современными масштабными языковыми моделями это требует значительных вычислительных ресурсов из-за большого количества параметров обучения. Идея метода снижения ранга заключается в том, чтобы эффективно адаптировать предварительно обученные модели к специфическим задачам, минимизируя количество параметров, которые необходимо «дообучить». Техника LoRA использует методы низкоранговой аппроксимации для уменьшения размерности весов модели и, таким образом, снижения вычислительной сложности и затрат памяти. Этот подход особенно полезен в условиях ограниченных вычислительных ресурсов для обучения модели или когда требуется быстрое приспособление к новым данным без полной перетренировки модели. LoRA позволяет сохранить ключевые контекстуальные представления, снижая при этом вычислительную нагрузку и требования к памяти. Для использования данной техники при обучении моделей была подключена библиотека peft, а также подключены такие библиотеки, как torch, datasets, transformers, numpy.

Следует отметить, что в процессе полного переобучения модели может возникнуть эффект «катастрофического забывания» [15], то есть модель может кардинально изменить свои характеристики. Таким образом, интересно оценить, насколько хорошо подвержены тонкой настройке различные модели, основанные на архитектуре RoBERTa, которые изначально не были обучены на текстах исходных кодов и модели, специально обученные на наборах данных из листингов программных кодов.

Задача выявления уязвимостей может быть естественным образом представлена как задача классификации фрагментов кода на классы с уязвимостями и без них. Для оценки точности моделей, в случае задачи классификации, используют такие показатели, как число истинно положительных прогнозов (True Positive – TP), число ложно положительных прогнозов (False Positive – FP), число истинно отрицательных прогнозов (True Negative – TN) и число ложно отрицательных прогнозов (False Negative – FN). Например, если классифицируются всего два класса, класс 1 и класс 0, то TP – число верно классифицированных примеров для класса 1, FP – число примеров, которые модель неверно отнесла к классу 1, TN – число примеров, которые верно отнесены к другим классам (класс 0), FN – показывает, число примеров, которые модель неверно отнесла к другим классам (классу 0).

В рамках исследования обучение моделей выполнялось с такими макропараметрами batch_size = 16, lora_alpha=32, lora_dropout=0.01, lr = 1e-3, task_type=’SEQ_CLS’. Кроме того, был выбран AdamW оптимизатор.

Для оценки адекватности полученной после тренировки модели была выполнена ее проверка на тестовой выборке для каждого класса.

Таблица 2

Показатели моделей на тестовой выборке

Модель	Число тренируемых параметров	TP	FP	TN	FN
RoBERTa-base	667 396	374	7	2475	1
CodeBERT-base	665 858	368	13	2473	3
RoBERTa-large	1 248 258	344	37	2476	0
RoBERTa-MLM-based PyTorrent	628 994	305	76	2474	2
CodeBERT-base-mlm	665 858	371	10	2474	2
CodeBERT-python	665 858	376	5	2476	0

Таблица 3

Показатели точности моделей

Модель	Полнота	Точность	F-мера
RoBERTa-base	0,997	0,981	0,989
CodeBERT-base	0,991	0,965	0,978
RoBERTa-large	1	0,902	0,948
RoBERTa-MLM-based PyTorrent	0,993	0,8	0,886
CodeBERT-base-mlm	0,994	0,973	0,984
CodeBERT-python	1	0,986	0,993

Оценка производилась путем записи в проверяемый файл записей только одного класса с последующим подсчетом количества прогнозов для каждого класса для получения значений TP, TN, FP, FN и вычисления метрик модели. В табл. 2 представлены полученные данные для различных моделей, а также общее число тренируемых параметров модели при ее тонкой настройке с использованием техники LoRA. Общее число записей в тестовой выборке для класса 1 составило 381 и 2476 для класса 0.

Поскольку классифицируемые классы имеют значительный дисбаланс, то для оценки моделей использовались показатели точности (Precision) и полноты (Recall) и F-мера.

$$precision = \frac{TP}{TP + FP},$$

$$recall = \frac{TP}{TP + FN}.$$

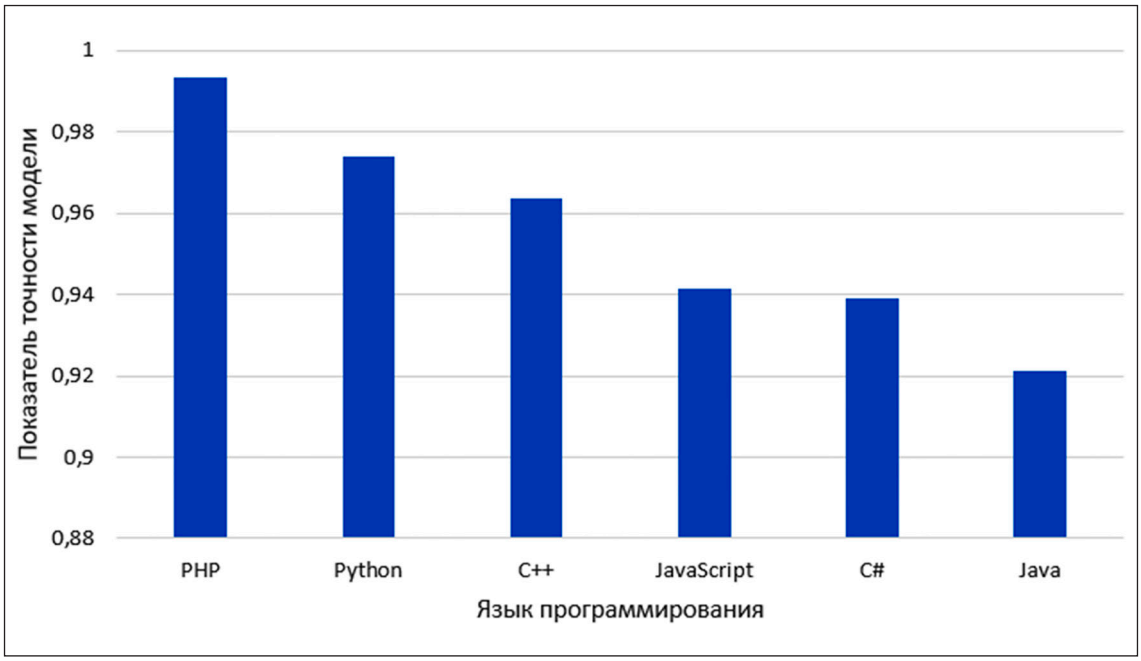
$$F = 2\frac{Precision \times Recall}{Precision + Recall}.$$

F-мера представляет собой среднее гармоническое точности и полноты. F-мера достигает максимума при полноте и точности, равными единице, и близка к нулю, если один из аргументов близок к нулю. В табл. 3 представлены значения полноты, точности и F-меры для анализируемых моделей.

Таким образом, получен ответ на вопрос В1 исследования. Так, лучшие показатели имеет модель CodeBERT-python, как по точности, так и по количеству ложно положительных значений. Данная модель может быть использована для дальнейшей тонкой настройки в качестве базовой модели для языка Python. В то же время показатели модели RoBERTa-base незначительно уступают CodeBERT-python. В связи с этим данная модель также может быть выбрана как базовая для дальнейшей тонкой настройки.

Для оценки адекватности адаптированной модели была выполнена проверка на синтетическом наборе данных, который был получен переводом тестовой выборки на различные императивные языки программирования, такие как C++, C#, PHP, Java, JavaScript. Показатели точности для данных языков представлены на рисунке.

Полученные данные свидетельствуют о высокой степени точности модели не только для языка Python, но и для других императивных языков программирования. Таким образом, получен положительный ответ на вопрос В2 о возможности использования адаптированных моделей в качестве универсального детектора жестко вписанных паролей для распространенных императивных языков программирования.



Показатели точности адаптированной модели для различных императивных языков программирования

Заключение

Исследование показало, что современные большие языковые модели могут эффективно использоваться для детектирования жестко закодированных паролей в программном коде. В работе были обучены ряд моделей, основанных на архитектуре Transformer. Для обучения моделей использовалась техника низкоранговой адаптации, которая позволяет значительно снизить вычислительные ресурсы для тонкой настройки модели. Все модели показали точность более 80 % и полноту более 90 %, на проверочной выборке данных, что свидетельствует о высокой степени адекватности моделей. На основании агрегированного показателя F-меры наилучшие результаты были получены при обучении модели CodeBERT-python. Таким образом, результат показывает, что для решения задачи детектирования жестко вписанных паролей в программном коде тонкая настройка данной модели наиболее эффективна. Кроме того, адаптированная модель показала высокие значения точности при детектировании жестко вписанных паролей и для выборок на отличных от Python императивных языках программирования. Таким образом, при решении прикладных задач статического анализа программного кода при выборе среди моделей, основанных на архитектуре RoBERTa, необходимо выбирать модели, обученные на максимально больших наборах данных для конкретного языка программирования. В случае отсутствия таких моделей в качестве базовой модели может быть выбрана модель RoBERTa-base, которая также показала высокие значения точности.

Список литературы

1. OWASP Top 10 – 2021. [Электронный ресурс]. URL: https://owasp.org/Top10/ (дата обращения: 22.04.2024).

2. Common Weakness Enumeration [Электронный ресурс]. URL: https://cwe.mitre.org/about/index.html (дата обращения: 22.04.2024).

3. Банк данных угроз безопасности информации. [Электронный ресурс]. URL: https://bdu.fstec.ru/vul (дата обращения: 24.04.2024).

4. Cousot P. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints // Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages. 1977. P. 238–252. DOI: 10.1145/512950.512973.

5. Welcome to the Bandit documentation! – Bandit documentation. [Электронный ресурс]. URL: https://bandit.readthedocs.io/en/latest/ (дата обращения: 22.04.2024).

6. Vaswani A., Shazeer N., Parmar N. et al. Attention is all you need // Advances in Neural Information Processing Systems. 2017. P. 5998–6008.

7. Edward H.J., Shen Y., Wallis P. et al. LoRA: Low-Rank Adaptation of Large Language Models // International Conference on Learning Representations. 2021. P. 152–178.

8. Hugging Face. Models. [Электронный ресурс]. URL: https://huggingface.co/models (дата обращения: 22.04.2024).

9. Liu Y., Ott M., Goyal N., Du J., Joshi M., Chen D., Levy O., Lewis M., Zettlemoyer L., Stoyanov V. RoBERTa: A Robustly Optimized BERT Pretraining Approach // arXiv.org. 2019. [Электронный ресурс] URL: https://arxiv.org/abs/1907.11692 (дата обращения: 26.04.2024).

10. Feng Zhangyin et al. CodeBERT: A Pre-Trained Model for Programming and Natural Languages // Findings. 2020. P. 1536–1547. DOI: 10.48550/arXiv.2002.08155.

11. Conneau A., Khandelwal K., Goyal N., Chaudhary V., Wenzek G., Guzmán F., Grave E., Ott M., Zettlemoyer L., Stoyanov V. Unsupervised Cross-lingual Representation Learning at Scale // Annual Meeting of the Association for Computational Linguistics. 2019. P. 8440–8451. DOI: 10.48550/arXiv.1911.02116.

12. Zhou S., Alon U., Agarwal S., Neubig G. CodeBERTScore: Evaluating Code Generation with Pretrained Models of Code // Conference on Empirical Methods in Natural Language Processing. 2023. P. 1325–1342. DOI: 10.48550/arXiv.2302.05527.

13. Bahrami M., Shrikanth N.C., Ruangwan S., Liu L., Mizobuchi Y., Fukuyori M., Chen W., Munakata K., Menzies T. PyTorrent: A Python Library Corpus for Large-scale Language Models // ArXiv.org. 2021. [Электронный ресурс]. URL: https://arxiv.org/abs/2110.01710 (дата обращения: 22.04.2024).

14. Kapustin D.A., Shvyrov V.V., Shulika T.I. Static Analysis of Corpus of Source Codes of Python Applications // Program Comput. Soft. 2023. № 49. P. 302–309. DOI: 10.1134/S0361768823040072.

15. McCloskey M., Neal J. Cohen. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem // Psychology of Learning and Motivation. 1989. Vol. 24. P. 109–165. DOI: 10.1016/S0079-7421(08) 60536-8.