Инженерный вестник Дона, №7 (2024)
ivdon.ru/ru/magazine/archive/n7y2024/9332

Применение языковых нейросетевых моделей для обнаружения вредоносного программного обеспечения

Д.М. Дудкин[1], *М.А. Кузнецов*[1], *Н.Г. Авдосев*[2], *В.А. Шабаловский*[1], *В.А. Егунов*[1]

[1] *Волгоградский государственный технический университет*
[2] *ООО «Яндекс. Технологии»*

Аннотация: Растущая популярность больших языковых моделей в различных сферах научной и индустриальной деятельности приводит к появлению решений, применяющих эти технологии для совершенно разных задач. В данной статье предлагается использовать языковые модели BERT, GPT и GPT-2 для обнаружения вредоносного программного кода. Предварительно обученная на естественных текстах нейросетевая модель дообучается на предобработанном датасете, содержащем программные файлы с вредоносным и безвредным кодом. Предобработка датасета заключается в том, что программные файлы в виде машинных инструкций транслируются в текстовое описание на формализованном языке. Дообученная таким образом модель используется для задачи классификации программного обеспечения на основе признака содержания в нем вредоносного кода. В статье приводится информация о проведенном эксперименте по использованию предложенной модели. Оценивается качество применения такого подхода в сравнении с существующими антивирусными технологиями. Предлагаются также пути улучшения характеристик модели.
Ключевые слова: антивирус, нейросеть, языковые модели, вредоносный код, машинное обучение, дообучение моделей, тонкая настройка, BERT, GPT, GPT-2.

Введение

В современную цифровую эпоху проблема обнаружения вредоносного программного обеспечения (ВПО) становится все более актуальной. Недобросовестные разработчики могут создавать вирусный код, который ставит под угрозу корректное функционирование любой цифровой системы от умного гаджета до большого вычислительного комплекса. Таким образом причиняется значительный вред как крупным корпорациям, так и конечным пользователям любых программируемых устройств. Необходимость анализа программного обеспечения (ПО) на наличие вредоносного кода становится критически важной процедурой [1]. Это следствие сложившейся тенденции распространения, установки и обновления ПО любого уровня через глобальную сеть. Так, по статистике «Лаборатории Касперского», в 2022

году более 15% компьютеров пользователей во всем мире хотя бы один раз подверглись атаке [2]. На рис. 1 отображено количество атакованных пользователей.

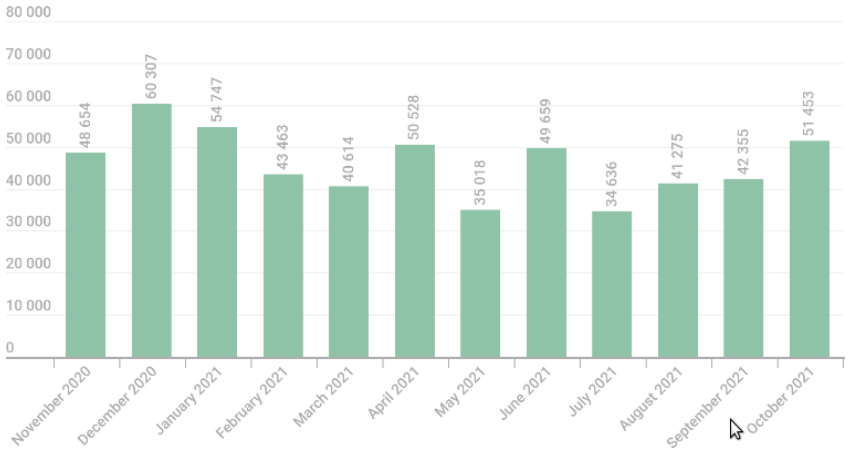

Рис. 1. – Количество пользователей, атакованных ВПО

Поскольку программные системы распространяются на различные сектора хозяйственной деятельности человека, риски применения ВПО постоянно растут. Киберпреступники ищут и находят новые уязвимости внутри существующих и вновь создаваемых программных систем. Значение анализа ПО для обнаружения и предотвращения использования вредоносных программ невозможно переоценить.

Принципы обнаружения ВПО

Существуют различные подходы, позволяющие выявлять ВПО. Эти подходы можно разделить на две большие основные группы: статические и динамические. Динамические методы опираются на поведенческие модели функционирования вредоносного кода, а статические методы основаны на выявлении специфических особенностей в последовательности кода.

Наиболее распространенным на практике подходом к обнаружению вредоносного ПО является сигнатурный метод. При его задействовании исполняемый код анализируется на соответствие определенному шаблону, соответствующего вредоносному коду. Причем при анализе сигнатур может

использоваться эвристические подходы. Сигнатурный анализ позволяет получить важные сведения о природе и характеристиках исследуемого ПО [3]. Важным преимуществом сигнатурных методов является высокая скорость и низкие требования к вычислительной производительности аппаратных устройств, на которых выполняется поиск ВПО [4]. Основным недостатком данного метода является гарантированное отсутствие сигнатур для нового или существенно модифицированного старого вредоносного кода. Задействование эвристики не исправляет данную проблему полностью. К тому же применение эвристических методов снижает точность определения ВПО.

Альтернативный динамический подход требует задействование специальных виртуальных машин, отслеживания вызовов прикладных API, обращений к системным библиотекам и функциям операционных систем. Анализ на ВПО осуществляется в процессе выполнения кода. Такой подход позволяет не только обнаруживать известное ВПО, но и выявлять аномальные действия в коде, что позволяет своевременно уведомлять об этом службу защиты. Для данных методов обнаружения ВПО характерны недостатки, связанные с высокими требованиями к вычислительной системе. Так как часть нагрузки вычислительной системы задействована на анализ работы кода параллельно с его выполнением, то наблюдается снижение производительности. Вопросы повышения точности обнаружения ВПО при задействовании эвристики также не решены полностью.

Задача поиска ВПО с математической точки зрения является задачей классификации исследуемых объектов, где объектом является программный код. В связи с бурным распространением методов Data Mining, машинного и глубокого обучения в последнее время при решении задач классификации все более часто задействуются нейронные сети. Создание обширных и доступных для анализа коллекций образцов вредоносных программ (таких как SoReL-

20M1, MalwareBazaar2) позволяет реализовать обучение нейронной сети и таким образом задействовать ее для решения задачи классификации программного кода по признаку вредоносности.

Описание используемого подхода

Программный код представляет собой описание алгоритма. Для процессора код реализуется в виде последовательности машинных инструкций. Однако алгоритм можно представлять и в более высокоуровневом виде – на одном из формальных языков программирования. Преимущество такого представления заключается в более абстрактном виде кода. К тому же код в виде текста имеет признаки естественного языка, что позволяет задействовать NLP методы, которые имеют довольно успешную историю применения на практике [5]. Кратко рассмотрим известные модели, используемые для обработки естественных языков.

BERT (Bidirectional Encoder Representations from Transformers) — это модель обработки естественного языка, разработанная Google. Она имеет 110 миллионов обучаемых гиперпараметров и использует двунаправленные трансформеры для создания глубоких контекстных представлений слов, учитывая контекст как слева, так и справа от целевого слова [6]. BERT обучается на больших объемах текстов, выполняя задачи маскирования слов и предсказания следующего предложения, что позволяет модели понимать смысл текста более полно.

Модели GPT (Generative Pre-trained Transformer) и GPT-2 разработаны OpenAI и представляют собой мощные инструменты для обработки и генерации естественного языка. GPT использует архитектуру трансформеров и обучается в два этапа: предобучение на большом объеме текста и дообучение на конкретных задачах [7]. GPT имеет 117 миллионов параметров, что позволяет ей выполнять различные задачи в области

обработки естественного языка. GPT-2 является улучшенной версией GPT и имеет значительно больше параметров — 1.5 миллиарда, что позволяет ей генерировать более связные и контекстуально осмысленные тексты [8]. Модель GPT-2 обучена на огромном корпусе данных и способна выполнять широкий спектр задач, включая перевод текста, создание осмысленных диалогов и написание статей.

Одним из популярных вариантов использования данных моделей является классификация текста. Данные модели хорошо показали себя в различных задачах классификации текста, такие как анализ настроения, классификация статей по категориям [9] и даже анализ действий человека [10].

Выдвинем следующую гипотезу. Представление машинного кода в виде текста на одном из формализованном языке программирования позволит анализировать текст алгоритма как текст естественного языка. Предпосылку к этой гипотезе дает основной принцип формальных языков программирования – стремление разработчиков таких языков сделать его близким к естественному языку. Наиболее простой способ перехода от машинного кода к коду на языке программирования — это процесс дизассемблирования. Текст на языке ассемблер довольно легко получить из машинного кода. Такой код уже имеет признаки естественного языка – определенную грамматику, приближенную к грамматике английского языка.

Обучение моделей

В качестве набора данных для обучения моделей был выбран набор [11], содержащий 10841 вредоносных файлов и 1082 – безвредных, состоящих из PE и OLE файлов. Он был предобработан следующим образом:

1) Оставлены только PE исполняемые файлы. При этом осталось 8933 вредоносных файла и 949 безопасных.

2) Исполняемые файлы были преобразованы в ассемблерные листинги. Удалены адреса инструкций и другая нехарактерная для естественных языков информация.

3) Предобработанный датасет разделен на три выборки:

– тренировочная: 50%,

– валидационная: 30%,

– тестируемая: 20%.

На тестируемой выборке проверяется качество моделей с помощью метрик, а первые две из них используются в обучении и валидации моделей между эпохами обучения.

Для обучения моделей использовался метод «тонкой настройки» (англ. fine-tuning), который представляет собой дообучение с разморозкой всех слоёв ранее предобученной модели. Предобученная модель ориентирована на анализ текстов на английском языке. Дообучение позволяет выполнить конкретную задачу классификации на основе признаков ВПО. Также для обучения были использованы адаптеры, позволяющие превратить модели в классификаторы. Для BERT на выход добавлен отвечающий за классификацию полносвязанный слой. А для задачи классификации с помощью GPT и GPT-2 используется последний смысловой токен из выхода.

Обучение происходило на рабочей станции, которая состояла из 16 ядерного процессора Intel Xeon, 256 гигабайт оперативной памяти и двух видеоускорителей NVIDIA RTX4090. Для программной реализации были выбраны язык Python и библиотека PyTorch.

В таблице 1 представлены параметры обучения моделей.

Таблица 1

Параметр	**Значение**
Скорость обучения	2e-5
Количество примеров на батч обучения	8

Количество примеров на батч валидации	8
Количество эпох	2
Weight decay	0.01

Полученные результаты

Для проверки качества полученных моделей используются 4 метрики:

– Accuracy — доля правильно предсказанных объектов среди всех объектов.

– Precision — доля правильно предсказанных положительных объектов среди всех, предсказанных как положительные.

– Recall — доля правильно предсказанных положительных объектов среди всех истинных положительных объектов.

– F1 Score — гармоническое среднее между precision и recall, которое учитывает оба этих показателя.

Все метрики принимают значение от 0 до 1, где 1 – наилучший результат. Они позволяют комплексно оценить качество моделей.

Также для сравнения были обучены классические модели машинного обучения, такие как «дерево решений» и «случайный лес». Но так как данные модели не способны принимать текст, было принято решение использовать модель TF-IDF, которая подготавливает текстовые данные для классических методов [12].

Как видно из данных, представленных в таблице 2, языковые модели хорошо показывают себя в задаче классификации кода на вредоносный и безвредный. При этом классические методы показали более низкие результаты.

Таблица 2

Модель / Метрика	**Accuracy**	**Precision**	**Recall**	**F1 Score**
BERT	0.97	0.98	0.99	0.98
GPT	0.95	0.95	0.99	0.97
GPT-2	0.98	0.98	0.99	0.99
Дерево решений	0.86	0.88	0.87	0.87
Случайный лес	0.88	0.90	0.88	0.86

Также для сравнительного анализа качества определения ВПО предложенным способом и активно используемыми существующими подходами датасет решили проверить на современных антивирусах с помощью сервиса VirusTotal [13]. Данный сервис предоставляет API для проверки файла на ВПО сразу на более чем 60 антивирусах. Лимиты бесплатного использования сервиса предполагают ограничения на количество вызовов, поэтому было проверено только около 1000 случайно выбранных из датасета файлов. В таблице 3 представлены результаты работы некоторых антивирусных программ в сравнении с полученной моделью на основе GPT-2.

Таблица 3

Название антивируса	**Accuracy**
Kaspersky	1
Dr. Web	0.996
Tencent	0.996
Google	0.995
Microsoft	0.991
Полученная модель на основе GPT-2	0.98

Yandex	0.841
Alibaba Cloud	0.661
Acronis	0.371

Часть существующих антивирусов показывает лучшие результаты, чем предложенная модель. Однако, если сравнивать затраты на поддержку работы данных антивирусов с довольно экономичным подходом при использовании языковых моделей, то соотношение качество/цена будет в пользу предложенного варианта. Кроме того, идея использования языковых моделей для обнаружения ВПО находится в самом начале своего развития. Она далеко не исчерпала своего потенциала. Увеличение качества работы возможно за счет следующих вариантов развития модели.

Во-первых, в рассмотренном эксперименте после дизассемблирования удалялась информация о переходах. Это выполнялось по причине того, что дизассемблер формировал численные значения адресов. Числа в тексте существенно отличали операторы ассемблера от структуры фраз естественного языка. Реализация переходов с определенными текстовыми ссылками формировала бы контекстную связь разных операторов, что несло бы дополнительную информацию об устройстве программного кода. С большой долей вероятности можно утверждать, что такая информация дополняет описание структуры кода. Это позволило бы более четко выделить особенности ВПО при дообучении языковой модели.

Во-вторых, для представления алгоритмов можно использовать вместо ассемблера специальный формализованный язык, который будет лучше передавать особенности существующих в коде алгоритмических структур. Такой язык должен лаконичнее показывать особенности кода, чем примитивные инструкции на языке ассемблер. Это также может увеличить качество определения ВПО.

Выводы

Проведенное исследование демонстрирует высокую эффективность использования больших языковых моделей (БЯМ) в задаче обнаружения вредоносного программного обеспечения. В работе рассмотрены модели BERT, GPT и GPT-2, которые показали значительно лучшие результаты в сравнении с классическими методами машинного обучения, такими как «дерево решений» и «случайный лес». Сравнение с современными антивирусными решениями показало, что предложенные модели могут быть конкурентоспособны.

Таким образом, использование БЯМ открывает новые перспективы в области кибербезопасности, предоставляя мощные инструменты для обнаружения и предотвращения запуска вредоносного программного обеспечения. Дальнейшие исследования могут быть направлены на улучшение моделей, увеличение объема тренировочных данных и адаптацию методов для интеграции с существующими системами безопасности.

Литература

1. Аникин И.В., Исяндавлетова Я.М. Реверсивный анализ вредоносного программного обеспечения Raccoon Stealer // Инженерный вестник Дона, 2023, №4. URL: ivdon.ru/uploads/article/pdf/IVD_57__4_Anikin_Isyandavletova.pdf_92265f4e3e.pdf.

2. Threat landscape for industrial automation systems. Statistics for H2 2022 // Kaspersky ICS CERT. URL: ics-cert.kaspersky.com/publications/reports/2023/03/06/threat-landscape-for-industrial-automation-systems-statistics-for-h2-2022/.

3. Souri A., Hosseini R. A state-of-the-art survey of malware detection approaches using data mining techniques // Human-centric Computing and

Information Sciences. 2018. Т. 8, № 1. С. 3. DOI: 10.1186/s13673-018-0125-x. URL: doi.org/10.1186/s13673-018-0125-x.

4. Canfora G., Iannaccone A.N., Visaggio C.A. Static analysis for the detection of metamorphic computer viruses using repeated-instructions counting heuristics // Journal of Computer Virology and Hacking Techniques. 2014. Т. 10, № 1. С. 11-27. DOI: 10.1007/s11416-013-0189-0. URL: doi.org/10.1007/s11416-013-0189-0.

5. Вакушин А.А., Клебанов Б.И. Применение больших языковых моделей в имитационном моделировании // Инженерный вестник Дона, 2024, №2. URL: ivdon.ru/uploads/article/pdf/IVD_72__1y24_vakushin_klebanov.pdf_afb02aa65c.pdf.

6. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Google AI Language. - 2019. - URL: arxiv.org/abs/1810.04805.

7. Radford A., Narasimhan K., Salimans T., Sutskever I. Improving Language Understanding by Generative Pre-Training. OpenAI, 2018. – URL: cdn.openai.com/researchcovers/languageunsupervised/language_understanding_paper.pdf.

8. Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I. Language Models are Unsupervised Multitask Learners. OpenAI, 2019. – URL: cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

9. González-Carvajal S., Garrido-Merchán E. Comparing BERT against traditional machine learning text classification // Journal of Computational and Cognitive Engineering. 2020. V. 2. DOI: 10.47852/bonviewJCCE3202838.

10. Zilelioglu H., Khodabandelou G., Chibani A., Amirat Y. Conditional Human Activity Signal Generation and Generative Classification with a GPT-2

Model // Proceedings of the International Joint Conference on Neural Networks (IJCNN). 2023. DOI: 10.1109/IJCNN54540.2023.10191464.

11. DikeDataset: Dataset with labeled benign and malicious files. URL: github.com/iosifache/DikeDataset (дата обращения: 08.06.2024).

12. Qaiser S., Ali R. Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents // International Journal of Computer Applications. 2018. DOI: 10.5120/IJCA2018917395.

13. VirusTotal API v3 Overview. URL: docs.virustotal.com/reference/overview (дата обращения: 08.06.2024).

References

1. Anikin I.V., Isyandavletova Ya. M. Inzhenernyj vestnik Dona, 2023, №4. URL: ivdon.ru/uploads/article/pdf/IVD_57__4_Anikin_Isyandavletova.pdf_92265f4e3e.pdf.

2. Threat landscape for industrial automation systems. Statistics for H2 2022. Kaspersky ICS CERT. URL: ics-cert.kaspersky.com/publications/reports/2023/03/06/threat-landscape-for-industrial-automation-systems-statistics-for-h2-2022/.

3. Souri A., Hosseini R. Human-centric Computing and Information Sciences. 2018. V. 8, № 1. p. 3. DOI: 10.1186/s13673-018-0125-x. URL: doi.org/10.1186/s13673-018-0125-x.

4. Canfora G., Iannaccone A.N., Visaggio C.A. Journal of Computer Virology and Hacking Techniques. 2014. T. 10, № 1. pp. 11-27. DOI: 10.1007/s11416-013-0189-0. URL: doi.org/10.1007/s11416-013-0189-0.

5. Vakushin A.A., Klebanov B.I. Inzhenernyj vestnik Dona, 2024, №2. URL: ivdon.ru/uploads/article/pdf/IVD_72__1y24_vakushin_klebanov.pdf_afb02aa65c.pdf.

6. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Google AI Language. 2019. URL: arxiv.org/abs/1810.04805.

7. Radford A., Narasimhan K., Salimans T., Sutskever I. Improving Language Understanding by Generative Pre-Training. OpenAI, 2018. URL: cdn.openai.com/researchcovers/languageunsupervised/language_understanding_paper.pdf.

8. Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I. Language Models are Unsupervised Multitask Learners. OpenAI, 2019. URL: cdn.openai.com/betterlanguagemodels/language_models_are_unsupervised_multitask_learners.pdf.

9. González-Carvajal S., Garrido-Merchán E. Journal of Computational and Cognitive Engineering. 2020. T. 2. DOI: 10.47852/bonviewJCCE3202838.

10. Zilelioglu H., Khodabandelou G., Chibani A., Amirat Y. Proceedings of the International Joint Conference on Neural Networks (IJCNN). 2023. DOI: 10.1109/IJCNN54540.2023.10191464.

11. DikeDataset: Dataset with labeled benign and malicious files. URL: github.com/iosifache/DikeDataset (date assessed 08.06.2024).

12. Qaiser S., Ali R. International Journal of Computer Applications. 2018. DOI: 10.5120/IJCA2018917395.

13. VirusTotal API v3 Overview. URL: docs.virustotal.com/reference/overview (date assessed: 08.06.2024).

Дата поступления: 10.05.2024

Дата публикации: 19.06.2024