УДК 004.492.3

НЕЙРОСЕТЕВОЙ ПОДХОД К ОБНАРУЖЕНИЮ ПРОГРАММНЫХ ДЕФЕКТОВ В ИСПОЛНЯЕМЫХ ФАЙЛАХ

Дудкин А.С. (ВКА)**, Молозаев А.Е.** (ВКА)
Научный руководитель – кандидат технических наук, доцент Дудкин А.С.
(Военно-космическая академия имени А.Ф.Можайского)

Введение. Современные программные системы становятся все более сложными и масштабными, что приводит к увеличению вероятности появления программных дефектов, которые могут варьироваться от незначительных ошибок, влияющих на пользовательский интерфейс, до серьезных уязвимостей, которые могут быть использованы злоумышленниками для нанесения ущерба. Традиционные методы обнаружения дефектов, такие как ручной анализ кода и статический анализ, часто оказываются недостаточно эффективными в выявлении сложных ошибок, особенно в больших кодовых базах. Кроме того, анализ исходного кода не всегда возможен, например, при работе с проприетарным программным обеспечением.

В последние годы наблюдается рост интереса к использованию машинного обучения для решения задач в области кибербезопасности, включая обнаружение программных дефектов. Нейронные сети, в частности, продемонстрировали впечатляющие результаты в различных областях, таких как обработка естественного языка и распознавание образов [1]. Применение нейронных сетей к анализу бинарных файлов представляет собой перспективный подход к автоматическому обнаружению уязвимостей и аномалий в исполняемых файлах. Этот подход позволяет анализировать низкоуровневые представления кода, не требуя исходного кода, что делает его применимым к широкому спектру программного обеспечения. В настоящей статье мы исследуем возможность использования различных архитектур нейронных сетей и методов обучения для создания эффективной системы обнаружения дефектов на основе анализа исполняемых файлов

Основная часть. Предлагаемый подход основан на использовании глубокого обучения для анализа бинарного кода, что позволяет обойти ограничения традиционных методов, которые часто требуют доступа к исходному коду или полагаются на заранее определенные правила. Глубокое обучение, в свою очередь, способно автоматически выявлять сложные и скрытые закономерности, указывающие на наличие дефектов, что делает возможным эффективную проверку как открытого, так и закрытого программного обеспечения [2].
Процесс анализа начинается с извлечения информативных признаков из бинарных файлов. Вместо того чтобы напрямую использовать последовательность байтов, что является малоэффективным, применяется многоуровневый подход. На низком уровне выделяются n-граммы байтов, которые отражают локальные контекстные зависимости, опкоды инструкций процессора, кодируемые в числовые векторы, и константы со строками, кодируемые с помощью word2vec [3] или fastText. Это помогает обнаруживать, например, зашитые пароли или SQL-запросы.

На среднем уровне создается граф потока управления (CFG), представляющий структуру программы в виде ориентированного графа, где узлы – это базовые блоки, а ребра – переходы управления. Аналогично формируется граф потока данных (DFG). Из этих графов извлекаются признаки, такие как степень узлов, длина путей и циклические структуры, которые представляются с помощью матрицы смежности или графовых эмбеддингов.

На высоком уровне используются результаты статического анализа, которые предоставляют информацию о типах данных, переменных и функциях, а также предварительно обученные эмбеддинги для улучшения качества представления признаков.

С целью анализа полученных признаков применяются различные архитектуры нейронных сетей. Сверточные сети (CNN) используются для анализа последовательностей байтов и n-грамм. Рекуррентные сети (RNN), такие как LSTM и GRU, анализируют

последовательности инструкций. Графовые нейронные сети (GNN) [4] обрабатывают CFG и DFG. Трансформеры применяются для обработки как последовательных, так и графовых данных. Многослойные персептроны (MLP) объединяют признаки, полученные от разных архитектур. Для повышения надежности применяется ансамблирование моделей.

Обучение происходит на размеченном наборе данных, содержащем исполняемые файлы с известными и неизвестными дефектами. Используются стандартные методы оптимизации и регуляризации [5]. Эффективность модели оценивается с помощью k-кратной перекрестной проверки и независимого тестирования. Для оценки измеряются точность, полнота, F1-мера, а также проводится анализ ложноположительных и ложноотрицательных результатов.

Выводы. Полученные результаты показывают, что предложенный подход демонстрирует многообещающие результаты обнаружения различных типов дефектов в автоматическом анализе исполняемых файлов, предоставляя возможность повышения безопасности программного обеспечения, а также демонстрируя перспективы применения машинного обучения для автоматического анализа бинарного кода.

Список использованных источников:

1. Очень глубокие сверточные сети для классификации текстов [Электронный ресурс]. – URL: https://arxiv.org/abs/1606.01781 – (дата обращения 09.01.2025).
2. ФОРМАТЫ ДВОИЧНЫХ ФАЙЛОВ [Электронный ресурс]. – URL: https://e-univers.ru/upload/iblock/86f/4qt13670wnwwfq9jrhn4sfpsy5taksv8.pdf?ysclid=m6nv1jr794572394875 – (дата обращения 01.01.2025).
3. NLP - Преобразование текста: Word2Vec [Электронный ресурс]. – URL: https://habr.com/ru/companies/otus/articles/574624/ – (дата обращения 09.01.2025).
4. Документация GNN [Электронный ресурс]. – URL: https://gnn-learning.readthedocs.io/_/downloads/en/latest/pdf/ – (дата обращения 09.01.2025).
5. Практическое руководство по передаче обучения для классификации текста с использованием сверточных нейронных сетей [Электронный ресурс]. – URL: https://arxiv.org/abs/1801.06480 – (дата обращения 09.01.2025).