УДК 33:004.431.4

ВЫЯВЛЕНИЯ УЯЗВИМОСТЕЙ В ДЕЦЕНТРАЛИЗОВАННЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ НА ОСНОВЕ СМАРТ-КОНТРАКТОВ С ПОМОЩЬЮ МЕТОДОВ ОБРАБОТКИ БОЛЬШИХ ДАННЫХ

Маркова С.В.

Финансовый университет при правительстве РФ, Москва, e-mail: Svmarkova@fa.ru

Смарт-контракты – это программы, которые хранятся в распределенном реестре и выполняют написанный в них код в ответ на адресованные им транзакции. Такие смарт-контракты написаны на языке программирования Solidity, который имеет определенную структуру и синтаксис. Язык был разработан для платформы Ethereum. Имея определенную структуру, такие языки подвержены определенным уязвимостям, использование которых может привести к большим финансовым потерям. В этой статье для обнаружения уязвимостей используется модель глубокого обучения (DL). Глубокое обучение – это тип алгоритма машинного обучения, основанного на технологии искусственной нейронной сети. Используя выбранный подход и правильно заданную структуру входных данных, можно обнаружить сложные зависимости между различными программными переменными, которые содержат уязвимости и ошибки. Этот подход был исследован с использованием набора определенных наборов данных, которые позволят изучить предлагаемую модель и улучшить ее производительность. Разработанная модель классифицировала уязвимости на уровне строк программного кода с использованием корпуса программных кодов смарт-контрактов Solidity в качестве входных данных. Применение модели DL позволяет выявлять уязвимости различной сложности в программном коде смарт-контрактов.

Ключевые слова: блокчейн, смарт-контракт, безопасность, уязвимость, глубокое обучение, децентрализованные приложения

IDENTIFYING VULNERABILITIES IN DECENTRALIZED INFORMATION SYSTEMS BASED ON SMART CONTRACTS USING BIG DATA PROCESSING METHODS

Markova S.V.

Financial University under the Government of the Russian Federation, Moscow, e-mail: Svmarkova@fa.ru

Smart contracts are programs that are stored in a distributed registry and execute code written in them in response to transactions addressed to them. Such smart contracts are written in the Solidity programming language, which has a specific structure and syntax. The language was developed for the Ethereum platform. Having a certain structure, such languages are subject to certain vulnerabilities, the use of which can lead to large financial losses. In this article, a deep learning (DL) model is used to detect vulnerabilities. Using the chosen approach and the correct input data structure, it is possible to detect complex dependencies between various program variables that contain vulnerabilities and errors. This approach has been investigated using a set of specific data sets that will allow you to study the proposed model and improve its performance. The developed model classified vulnerabilities at the level of lines of program code, using the corpus of program codes of Solidity smart contracts as input data. The use of the DL model makes it possible to identify vulnerabilities of varying complexity in the software code of smart contracts.

Keywords: blockchain, smart contract, security, vulnerability, deep learning, decentralized applications

Уязвимость программного обеспечения – это «наличие недостатка в конструкции, слабости или реализации, которые могут привести к нежелательному событию, ставящему под угрозу работу программного приложения, компьютерной системы, сети или протокола» [1].

Поскольку популярность смарт-контрактов и количество связанного с ними программного обеспечения увеличиваются с каждым днем, то прямо пропорционально увеличивается количество уязвимостей и атак, влияющих на работу этого программного обеспечения.

Децентрализованные информационные системы на основе смарт-контрактов необходимо защищать, чтобы предотвратить ошибки в программном коде и нежелательные последствия, которые могут возникнуть из-за них. Примерами таких ошибок являются проблемы уязвимости децентрализованных приложений. Так, из-за ошибки в рекурсивном вызове функций в программном коде проекта The DAO было потеряно более 50 млн долларов [2]. Ущерб от срабатывания уязвимостей – одна из причин их детального и тщательного расследования. Некоторые из причин уязвимостей – наличие сложных взаимодействий между частями программного кода и несоответствие между тем, что программа должна делать, и тем, что она делает на самом деле. Поскольку невозможно гарантировать отсутствие уязвимостей при написа-

нии программного кода смарт-контрактов, то актуальными являются исследования, связанные с разработкой методов и моделей обнаружения уязвимостей в программном коде смарт-контрактов.

В настоящее время существуют инструменты [3, 4, 5, 6] для статистического анализа, распознающие ошибки и уязвимости. Главный недостаток таких инструментов состоит в том, что они позволяют обнаружить лишь ограниченное количество ошибок на основе заранее определенных правил. Для понимания важности и опасности обнаруженных уязвимостей необходим дополнительный анализ специалистом в данной предметной области. Это часто приводит к ошибочным выводам при оценке смарт-контрактов на наличие уязвимостей. Кроме того, несмотря на наличие большого количества литературы, в которой описаны основные уязвимости и методы их обнаружения, все еще существуют классы уязвимостей и ошибок, которые очень трудно или невозможно распознать [7].

В данной работе предлагается методика выявления уязвимостей в программном коде смарт-контрактов децентрализованных приложений, которая основывается на использовании алгоритмов глубокого обучения на больших данных (Big Data) в виде открытых исходных кодов смарт-контрактов. Использование большого объема входных данных, находящихся в открытых репозиториях, даст возможность обнаружить большее количество уязвимостей. В качестве исходных данных используется корпус из программных кодов смарт-контрактов, анализ которых позволяет выявить глобальные закономерности проявления ошибок и уязвимостей, что невозможно сделать статической проверкой.

Цель исследования: разработать методику, которая поможет выявить уязвимости различной сложности, возникающие при работе смарт-контрактов, что позволит более точно определить, является ли смарт-контракт безопасным и пригодным для исполнения или нет.

В работе предлагается модель на основе алгоритмов машинного обучения, которая с помощью методов бинарной классификации определяет, содержит ли смарт-контракт уязвимости и ошибки. В качестве входных данных использовался корпус программ смарт-контрактов, разработанный на языке программирования Solidity. Корпус смарт-контрактов рассматривался как абстрактное синтаксическое дерево (AST). Предлагаемая модель была исследована на различных вариантах исходных данных для определения ее реакции на различные типы входных данных.

Материалы и методы исследования

Исследования по обнаружению ошибок и исправлению программного обеспечения в последнее время стали важной областью изучения и анализа [8]. С появлением новых языков программирования и новых областей их применения, таких как разработка децентрализованных приложений на основе технологии блокчейн и смарт-контрактов, проблема выявления уязвимостей и ошибок в программном коде становится все более актуальной. Для того чтобы повысить точность выявления уязвимостей и ошибок в программном коде, необходимо решить две основные проблемы. Во-первых, представление исходного кода, используемого в качестве входных данных, должно отражать внутреннюю структуру программы и взаимосвязь между переменными в ней. Во-вторых, модель должна быть спроектирована так, чтобы в полной мере использовать предлагаемую структуру входных данных.

Существует ряд исследований, которые предлагают подходы и методы для решения данных проблем. Так, в работе [9] был предложен метод обнаружения уязвимостей, который показал, что обучение на искусственно созданных ошибках позволяет получать детекторы ошибок, которые эффективны при обнаружении ошибок в реальном коде. Другое решение – использовать существующие средства обнаружения ошибок, как это было рассмотрено в работе [10].

В работе [11] был предложен автоматизированный подход к изучению характеристик повторяющегося кода смарт-контрактов. Данный подход позволяет выявлять ошибки в программном коде и выполнять проверку кода смарт-контракта на наличие уязвимостей. В работе [12] применяются алгоритмы и методы глубинного обучения (DL) для того, чтобы воспользоваться особой структурой исходного кода. В указанной работе предлагается модель классификации смарт-контрактов на основе технологии машинного обучения, называемой долговременной кратковременной памятью (LSTM).

В процессе исследования и анализа существующих уязвимостей в смарт-контрактах были выявлены наиболее распространенные типы уязвимостей [10–12]:

− целочисленное переполнение;

− возвращаемое значение непроверенного вызова;

− состояние исключения (инвариантное утверждение).

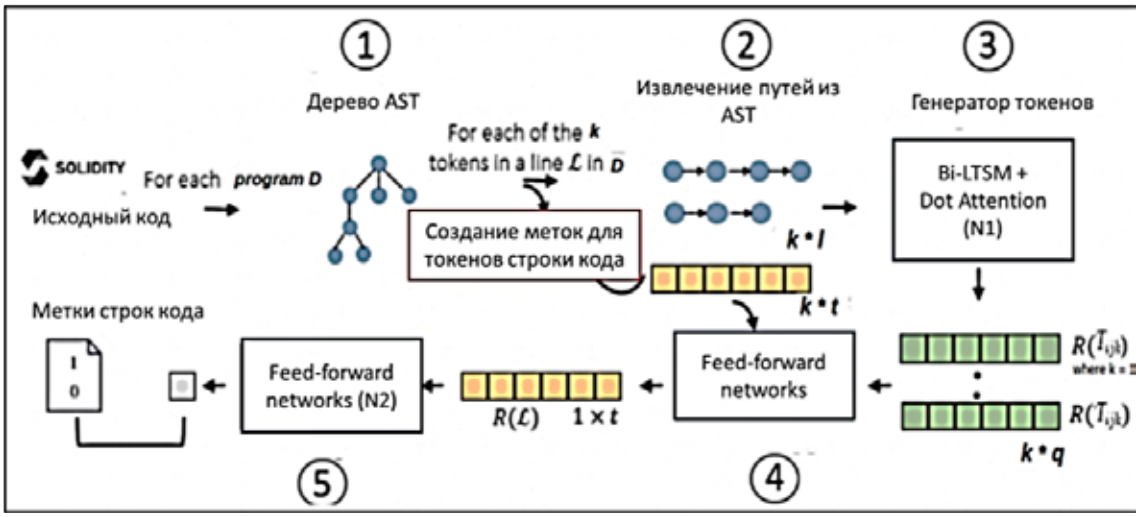

Рис. 1. Предлагаемая модель оценки уязвимостей смарт-контрактов

В работе предлагается методика по обнаружению уязвимостей в смарт-контрактах, которая включает в себя следующие этапы:

– создание корпуса из программных кодов смарт-контрактов Solidity, который используется в качестве исходных данных;

– представление корпуса в виде AST-дерева для понимания сложных зависимостей между переменными в программах;

– тестирование модели с использованием различных типов входных данных;

– реализация модели глубинного машинного обучения (DL) для обнаружения паттернов, вызывающих уязвимости;

– исследование полученных моделей в различных экспериментальных условиях;

– использование механизма внимания (Dot Attention) в модели DL.

Архитектура предложенной модели для классификации уязвимостей в программном коде смарт-контрактов представлена на рисунке 1. Модель принимает корпус контрактов Solidity в качестве входных данных. На основе этих данных определяются метки для каждой строки программного кода, которые используются при бинарной классификации для обнаружения уязвимостей в программном коде. Представление пути AST применяется для создания одного пути для каждого токена, принадлежащего каждой строке для каждого контракта. Эти действия представлены шагами 1 и 2.

Затем, на шаге 3, данные, полученные после шага 2, используются в качестве входных данных для LSTM-сети вместе с Dot Attention. Непротиворечивые свойства текста были учтены посредством использования LSTM-сети [12]. В разработанной модели уровень внимания (Dot Attention), предложенный в работе [13], играет ключевую роль в процессе обучения, а также является строительным блоком интерпретируемости предлагаемой модели. Цель Dot Attention – имитировать механизм человеческого внимания. Этот шаг делает модель более способной к изучению контекста и отношений между токенами в коде.

Конкретная сеть, использующая эту концепцию внимания, определена как N1, используется на шаге 3 и создает вектор внедрения на уровне токенов, который объединяется с информацией о конечных точках для передачи в простую сеть прямого распространения. Вложения на уровне токенов, созданные на шаге 3, являются распределенным представлением измерения и содержат информацию, соответствующую путям управляющих данных. Представления каждого токена, образующего определенную линию, объединяются. Эта конкатенация затем используется для создания линейного представления использования прямой связи для сети, определенной на шаге 4.

Таким образом, это последнее встраивание на уровне строк используется в другой сети прямого распространения, чтобы уменьшить размер вектора и, наконец, получить по одной бинарной метке на строку. Сеть N2 изучает взаимосвязь между строкой кода и присвоенной меткой. Статистический анализ весов внимания позволяет понять закономерности, присущие уязвимостям. Таким образом, модели в основном нужны два источника информации: пути AST на уровне токенов и сохраненные индексы конечных точек. Они создают в моделях два набора включений: включения

на уровне токенов и включения на уровне строк. Для создания последнего необходимы чередующиеся конечные точки уровня строки. При сопоставлении с методами обработки естественного языка (NLP) предлагаемую модель можно сравнить с генерацией вложений для каждого предложения и, в конечном итоге, каждого абзаца, поскольку ввод NLP состоит из нескольких токенов на строку, а абзац – это набор предложений. В этом случае встраивание на уровне абзаца используется для отображения уязвимостей на уровне строки.

Результаты исследования и их обсуждение

Выполним оценку обучаемости предложенной модели классификации уязвимостей программного кода смарт-контрактов.

Входными данными для обучения и тестирования модели являются представления программного кода смарт-контрактов. Эти входные данные были случайным образом разделены на 3 набора: тестовый набор, соответствующий 30% всего набора данных; остальные 70% были дополнительно разделены на наборы данных для обучения и проверки: таким образом, набор для обучения соответствует 49%, а набор для проверки – 21% всего набора данных. Потери, используемые во время обучения, представляют собой взвешенную кросс-энтропийную потерю, выбранную из-за ее способности справляться с несбалансированными входными данными. Наши модели были реализованы с использованием PyTorch версии 1.0.

Как и ожидалось для работающей модели DL, затраты (потери) во время обучения уменьшаются, затраты проверки ведут себя аналогично. Анализ графиков на рисунке 2 показывает отсутствие переобучения предложенной модели, так как кривая обучения имеет ожидаемую форму.

Последние два графика (Metrics и Rates) на рисунке 2 показывают эволюцию различных показателей, используемых для оценки производительности предложенной модели в разные периоды обучения.

Графики, представленные на рисунках 2 и 3, показывают, что предложенная модель способна понимать входные данные и может изучать статистические закономерности, чтобы без проблем классифицировать строки программного кода, которые содержат уязвимость.

Рисунок 3 предсказывает вероятности для каждого класса. Однако вероятности можно интерпретировать с использованием разных пороговых значений. Изменение этого параметра может изменить производительность. Первый график на рисунке 3 показывает соотношение TPR и FPR.

Кривая рабочих характеристик (ROC) иллюстрирует компромисс между обеими метриками для прогнозной модели с использованием различных пороговых значений правдоподобия. На втором графике показаны показатели F1, точности и отзыва в зависимости от порогового значения.

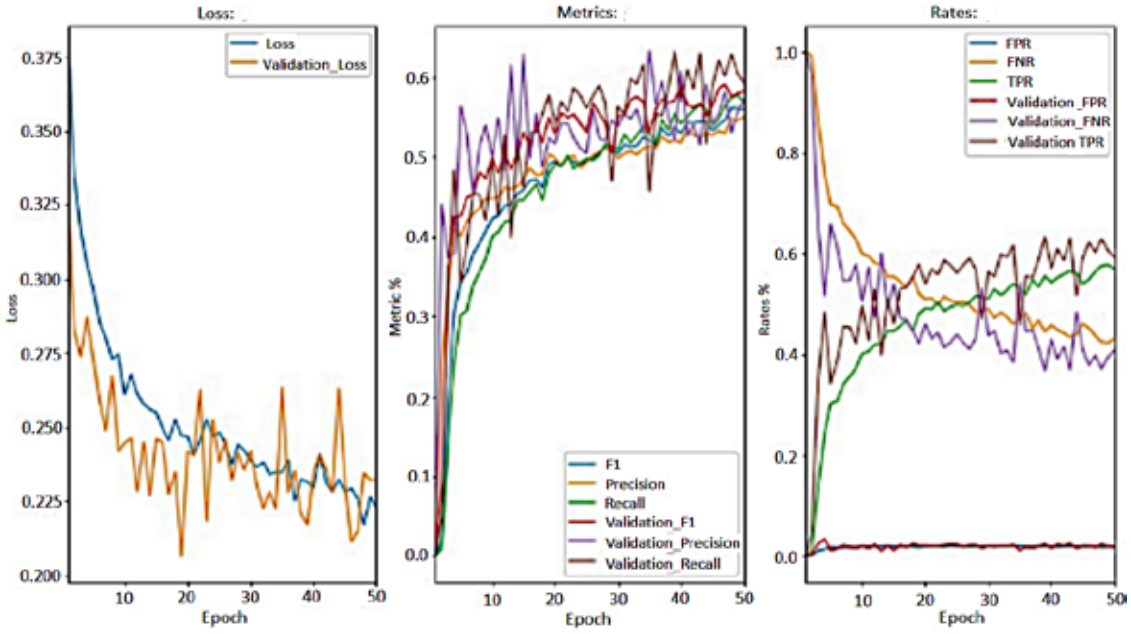

Рис. 2. Кривые обучения и оценки на этапе обучения модели для обучающей и тестовой выборки

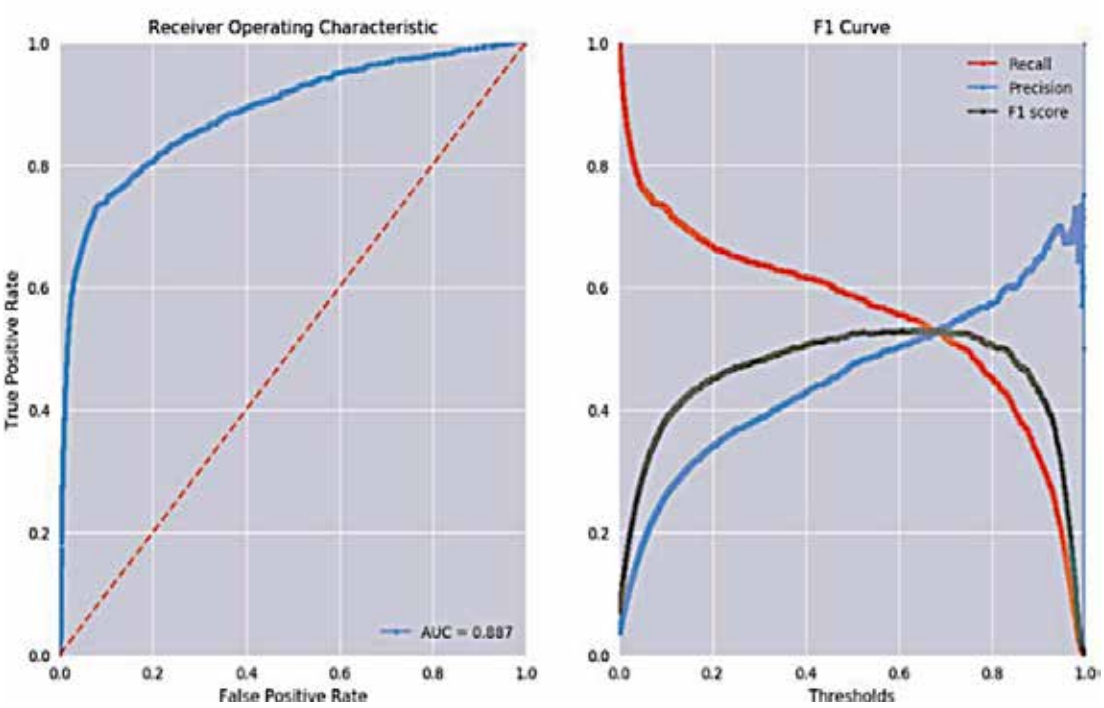

Рис. 3. ROC-кривая модели и кривые Scores, определенные в зависимости от порога классификации

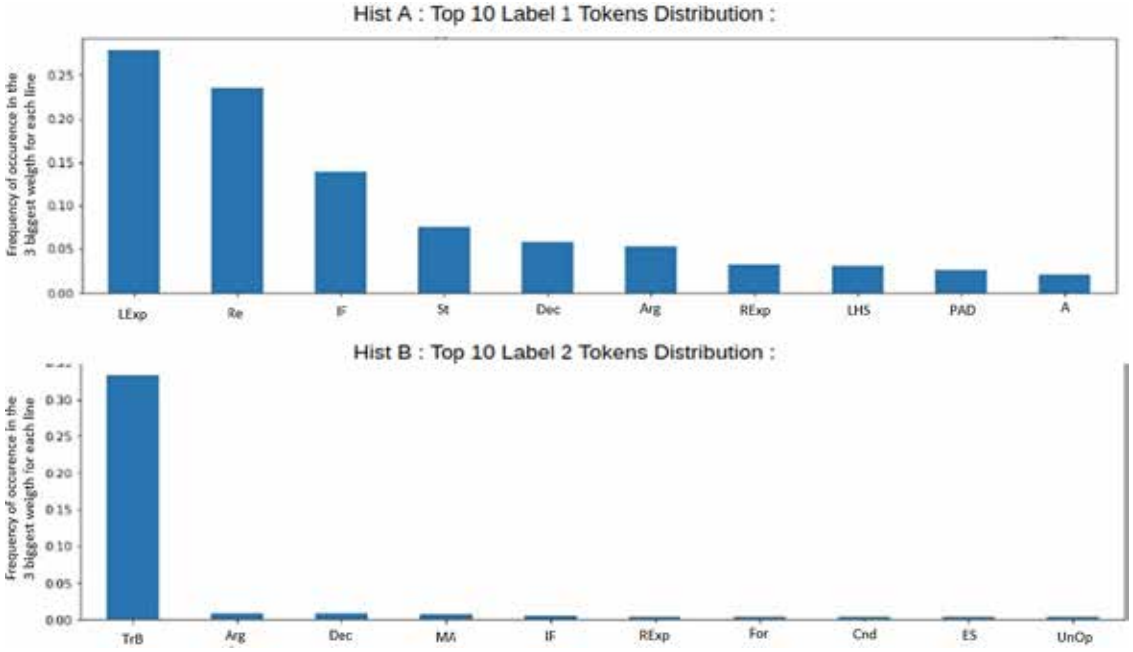

Рис. 4. Результаты анализа веса внимания на гистограмме A и гистограмме B

Графики, представленные на рисунках 2 и 3, доказывают, что предложенная модель способна понимать входные данные и может изучать статистические шаблоны для классификации строк программного кода с уязвимостями.

Гистограммы A, B, C представляют относительную важность токенов Top10, используемых для каждого типа уязвимости (рис. 4, 5). Это означает, что из этих графиков можно получить общее представление о том, какие токены описывают какие типы уязвимостей.

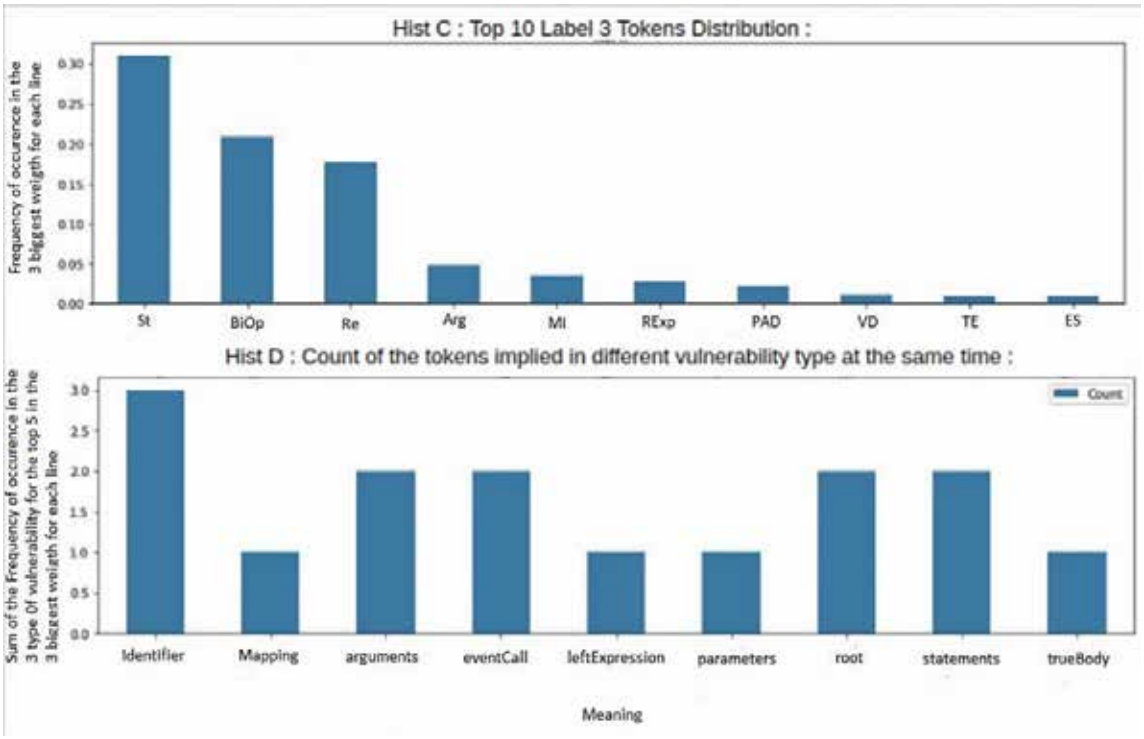

Рис. 5. Результаты анализа веса внимания на гистограмме C и гистограмме D

Таким образом, эти графики суммируют причины каждого типа уязвимости, что позволяет интерпретировать модель.

Гистограмма D отображает счетчик, соответствующий типу уязвимости для каждого токена (рис. 5). Этот счетчик показывает, сколько раз каждый токен был задействован в уязвимости. Это означает, что максимальный рейтинг, который может иметь токен, равен 3, из этого следует, что этот токен является одной из причин трех различных типов уязвимостей, изученных одновременно.

Если счетчик равен 2, это означает, что токен присутствует в двух типах уязвимостей, поэтому он может подразумеваться в типах 1 и 2, или типах 1 и 3, или типах 2 и 3. Кроме того, выполняя один и тот же анализ только для двух типов уязвимостей, можно определить некоторые общие причины для обоих типов: более важные общие причины типов 1 и 2 – это операторы и узлы вызова событий типов 1 и 3.

Выводы

Эта работа представляет научный интерес в вопросах обнаружения уязвимостей в программном коде децентрализованных приложений, написанных на Solidity. Предлагаемая модель фиксирует сложные зависимости управления и данных и успешно классифицирует 3 типа уязвимостей. Информация о конечных точках была подтверждена экспериментальным путем с использованием синтетических и необработанных данных. Это позволяет лучше понять естественную структуру программного кода смарт-контракта. Информация из программных токенов, хотя семантически не способна фиксировать уязвимости, повышает точность моделей. Интерпретируемость модели была повышена за счет использования механизма внимания (Dot Attention).

Список литературы

1. European Union Agency for Cybersecurity. Vulnerability definition, enisa.europa.eu. https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/glossary

2. Begum A., Hasnat Tareq A., Sultana M., Khaled Sohel M., Kahman T., H.Sarower A., Bhuiyan T. Blockchain attacks analysis and a model to solve double spending attack. International Journal of Machine Learning and Computing. 2020. T. 10. №. 2. P. 352-357.

3. Wang A., Wang H., Jiang B., Chan W. K. Artemis: An improved smart contract verification tool for vulnerability detection. 7th International Conference on Dependable Systems and Their Applications (DSA). IEEE. 2020. P. 173-181.

4. Praitheeshan P., Pan L., Doss R. Security evaluation of smart contract-based on-chain Ethereum wallets. International Conference on Network and System Security. Springer, Cham. 2020. P. 22-41.

5. Leid A., van der Merwe B., Visser W. Testing Ethereum smart contracts: A comparison of symbolic analysis and fuzz testing tools. Conference of the South African Institute of Computer Scientists and Information Technologists. 2020. P. 35-43.

6. Durieux T., Abreu R., Ferreira J.F., Gruz P. Empirical review of automated analysis tools on 47,587 Ethereum smart contracts. Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. 2020. P. 530-541.

7. Tang X., Zhou K., Cheng J., Li H. The Vulnerabilities in Smart Contracts: A Survey. International Conference on Artificial Intelligence and Security. Springer. Cham. 2021. P. 177-190.

8. Xing C., Chen Z., Chen L., X Guo X., Zheng Z., Li J. A new scheme of vulnerability analysis in smart contract with machine learning. Wireless Networks, 2020. P. 1-10.

9. Harer J., Kim L., Russell R, Ozdemir O., Kosta L., Rangaman A., Hamilton L., Centeno G., Key J., Ellingwood P., Antelman E., Mackay A., McConley M., Opper J., Chin P., Lazovic T. Automated software vulnerability detection with machine learning. arXiv preprint arXiv:1803.04497. 2018.

10. Yamashita K., Nomura Y., Zhou E., Pi B., Jun S. Potential risks of hyperledger fabric smart contracts. IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE). IEEE. 2019. P. 1-10.

11. Gao Z., Jiang L., Xia X., Lo D., Grundy J. Checking smart contracts with structural code embedding. IEEE Transactions on Software Engineering. 2020.

12. Goginen A., Swayamjyoti S., Sahoo D., Sahu K., Kishore R. Multi-Class classification of vulnerabilities in Smart Contracts using AWD-LSTM, with pre-trained encoder inspired from natural language processing. IOP SciNotes. 2020. T. 1. No 3. P. 035002.

13. Kakanakou M., Xie H., Qiang Y. Double attention mechanism for sentence embedding. International Conference on Web Information Systems and Applications. Springer. Cham. 2018. P. 228-239.