**DOI: https://doi.org/10.23670/IRJ.2021.103.1.011**

# ИССЛЕДОВАНИЕ БЕЗОПАСНОСТИ СМАРТ-КОНТРАКТОВ ETHEREUM

Научная статья

**Команов П.А.[1, *], Ревазов Х.Ю.[2], Тавасиев Д.А.[3]**

[1] ORCID: 0000-0001-5640-2322;

[2] ORCID: 0000-0002-3358-6197;

[3] ORCID: 0000-0001-7859-9461;

[1, 2, 3] Национальный исследовательский университет информационных технологий, механики и оптики, Санкт-Петербург, Россия

* Корреспондирующий автор (pavel.komanov[at]yandex.ru)

**Аннотация**

В настоящее время технологии блокчейн находят высокий спрос в разных сферах. Широкое применение находят и смарт-контракты, которые записываются в блокчейн. Наибольшую популярность на данный момент имеют приложения, написанные с помощью смарт-контрактов на платформе Ethereum. Смарт-контракты, как и обычные программы, подвержены различным уязвимостям. В статье будут рассмотрены серьезные уязвимости, методы и инструменты, которые были менее освещены в сферах информационной безопасности и IT, но имеющие огромный потенциал в области обеспечения безопасности смарт-контрактов. Основная цель данной статьи заключается в том, чтобы дать представление об актуальных и серьезных уязвимостях в смарт-контрактах Ethereum и на основе актуальных угроз подобрать актуальные методы по обеспечению безопасности смарт-контрактов на платформе Ethereum.

**Ключевые слова:** смарт-контракт, уязвимости, блокчейн, Ethereum.

# A SECURITY STUDY OF ETHEREUM SMART CONTRACTS

Research article

**Komanov P.A.[1, *], Revazov H.Yu.[2], Tavasiev D.A.[3]**

[1] ORCID: 0000-0001-5640-2322;

[2] ORCID: 0000-0002-3358-6197;

[3] ORCID: 0000-0001-7859-9461;

[1, 2, 3] ITMO University, St. Petersburg, Russia

* Corresponding author (pavel.komanov[at]yandex.ru)

**Abstract**

Currently, blockchain technologies are in high demand in various fields. Smart contracts that are stored in the blockchain are also widely used. The most popular applications at the moment are those made with the help of smart contracts on the Ethereum platform. Smart contracts, like ordinary programs, are subject to various vulnerabilities. This article discusses serious vulnerabilities, methods and tools that have been less covered in the areas of information security and IT, but have a huge potential in the field of smart contract security. The main purpose of this article is to give an idea of the current and major vulnerabilities of Ethereum smart contracts and, based on the threats, select relevant methods for ensuring the security of smart contracts on the Ethereum platform.

**Keywords:** smart contract, vulnerabilities, blockchain, Ethereum.

## Введение

Одной из самых популярных блокчейн платформ по состоянию на 6 апреля 2020 года, исходя из текущей капитализации рынка криптовалют, является Ethereum. Сеть Ethereum способна размещать алгоритмы, выраженные на языке программирования общего назначения. Это позволяет разработчикам создавать множество приложений, от простых кошельков до финансовых систем, новых криптовалют или систем торговли энергией. Вместо того, чтобы создавать отдельный блокчейн для каждого приложения, можно реализовать множество вариантов использования приложений с помощью технологии, известной как смарт-контракты.

Смарт-контракт представляет собой фрагмент кода, хранящийся в блокчейне. Он приводится в действие транзакциями и имеет возможность читать данные или писать их в блокчейн, а также позволяет создавать свои токены, которые работают на платформе Ethereum. Код контракта пишется на языке программирования Solidity [1].

Язык программирования общего назначения на платформе блокчейн дает возможность для реализации широкого спектра децентрализованных приложений. Однако, с другой стороны, такие приложения дают возможность злоумышленникам использовать их в своих корыстных целях. В этой работе исследуем известные и обновленные уязвимости в смарт-контрактах платформы Ethereum. Также в работе исследуем методы и инструменты анализа кода безопасности, используемые для выявления уязвимостей и ошибок в смарт-контрактах. Насколько известно, данная работа будет уникальным вкладом в данную область, поскольку анализ существующих инструментов не проводился.

## Уязвимости в смарт-контрактах Ethereum

В данном разделе будет дано краткое объяснение каждой уязвимости системы безопасности. Было принято опустить небольшие очевидные проблемы в смарт-контрактах Ethereum и сделать основной упор на серьезные уязвимости в данной системе.

### Уязвимость повторного входа

Повторный вход считается серьезной уязвимостью, так как была признана крупнейшей атакой из когда-либо совершенных (взлом от 18 июня 2016 года на The DAO) [2]. Принцип данной уязвимости заключается на

взаимодействии между двумя смарт-контрактами, один из которых является контрактом пользователя (X) и мошеннический контракт (Y). Контракт (X) отслеживает несколько внешних адресов для получения баланса и извлекает контрольную сумму. Контракт (Y) начинает взаимодействовать с контрактом (X), и тем самым контракт (X) передает управление контракту (Y). Контракт (X) выполняет функцию внешнего вызова, для отправки некоторого количества эфира в контракт (Y). При этом контракт (Y) может получить несколько возвратов и очистить баланс контракта (X).

Уязвимость повторного входа можно предотвратить, обеспечив фиксацию логики изменения состояния до того, как эфир будет отправлен из контракта посредством внешнего вызова. Еще одним отличным способом может служить, использование мьютекса путем добавления переменной состояния, которая блокирует контракт во время выполнения кода, предотвращая повторные вызовы функций.

**Неверно обработанные исключения**

Существует много ситуаций, когда исключение может быть вызвано в Solidity, но сам способ обработки этих исключений не всегда одинаков. Обработка исключений основана на взаимодействие между контрактами [3]. Тем самым контракты уязвимы для атак со стороны злоумышленников, если эти исключения не будут обработаны должным образом, то транзакции будут отменены.

**DoS, вызванная из-за уязвимости внешнего вызова**

Отказ в обслуживании включает в себя достижение предела газа или ограниченным лимитом газа [4]. Данная уязвимость имеет высокую оценку критичности по сравнению с остальными для платформы Ethereum, в то время как другие типы приложений могут восстановиться, смарт-контракты могут быть отключены навсегда всего лишь одной из этих атак.

DoS, вызванная из-за уязвимости внешнего вызова, обусловлена тем, что когда поток управления передается внешнему контракту, выполнение контракта вызывающего абонента может случайно или преднамеренно завершиться неудачей, что и вызывает DoS в контракте вызывающего абонента. Также отказ в обслуживании может быть вызван, когда вызываемый контракт намеренно вызывает откат транзакции, чтобы нарушить выполнение контракта вызывающего абонента.

**Безгазовая отправка (Gasless send)**

Уязвимость типа «безгазовая отправка» приводит к сбою транзакции, если для конкретного вызова не хватает газа [2]. Газ – внутренняя валюта, которую пользователь должен заплатить для вызова смарт-контракта. Количество газа зависит от сложности вычислений в смарт-контракте, чем сложнее вычисления, тем больше газа придется заплатить. Максимальный лимит газа в сети может меняться со временем в зависимости от комиссии за транзакцию.

Для предотвращения данной уязвимости важно создать исключение, если произойдет сбой из-за расхода газа. Кроме того, важно разработать функции, которые не требуют слишком большого количества газа не только с целью предотвращения сбоя, но и с единственной целью снижения затрат на выполнение контракта.

**Зависимость от временной метки**

Если контракт использует данную функцию для критической проверки, майнер может манипулировать меткой времени в течение нескольких секунд, при этом изменяя вывод средств в свою пользу [4]. Однако эта уязвимость является серьезной только в том случае, если она используется в критических проверках компонентов контракта.

**Использование blockhash функции**

Использование blockhash функции аналогично зависимости временной метки, ее не рекомендуется использовать для важных компонентов по той же причине, что и с зависимостью от временной метки, потому что майнеры могут манипулировать данными функциями и изменять вывод средств в свою пользу. Особенно это заметно, когда blockhash используется как источник случайности.

**Методы и инструменты по обеспечению безопасности смарт-контрактов**

Для того чтобы обнаружить уязвимости в коде приложения, большинство разработчиков приложений используют стандартные правила по их нахождению. При тестировании на выявление уязвимостей в приложениях используется различные методы, в основном это бывают статические и динамические анализаторы кода. В данной статье речь пойдет о фаззинг тестировании [5]. Самым популярным фаззингом считается метод полупрозрачного ящика или тестирование серого ящика. Яркими представителями таких фаззеров являются AFL и AFL Fast. Метод серого ящика предоставляет доступ к внутренней структуре и алгоритмам работы программного обеспечения для дальнейшего написания тест-кейсов [6]. Такое тестирование с точки зрения смарт-контрактов зачастую не эффективно.

ContractFuzzer – это новейший метод фаззинг тестирования черного ящика для смарт-контрактов [7]. Фаззер генерирует тестовые входные данные на основе двоичного интерфейса приложений (ABI) тестируемого смарт-контракта и формирует набор из семи тестовых оракулов (Тест Оракула) [8] для обнаружения следующих уязвимостей: безгазовая отправка, повторный вход, зависимость от временной метки, уязвимость внешнего вызова. Фаззер использует EVM [2] для регистрации и отслеживания поведения смарт-контрактов по случайным сгенерированным транзакциям, анализирует журналы для сообщения об уязвимостях безопасности [5]. Для поддержки некоторых тестовых оракулов фаззер генерирует дополнительные контракты при нечетком тестировании смарт-контрактов.

Если требуется провести анализ безопасности в области смарт-контрактов, в которых присутствуют уязвимости похожие или затрагивающие газ, такие как «безгазовая отправка» или «DDoS атака с ограниченным лимитом газа» [4], необходимо применять метод тестирования GasFuzzer [9]. Фаззер имеет несколько стратегий повышения эффективности обнаружения уязвимостей безопасности. Одна из стратегий GasFuzzer заключается в том, чтобы

отдавать приоритет транзакциям, которые потребляют больше газа, чем другие, для мутации входных параметров. Немного о стратегии тестирования GasFuzzer. GasFuzzer изначально генерирует транзакции случайным образом, подобно ContractFuzzer, и выполняет тестирование смарт-контрактов с этими транзакциями. Далее транзакции видоизменяются и отправляются в блокчейн. Если новые сгенерированные транзакции снова потребляют больше газа, они добавляются в очередь ввода для дальнейших возможных изменений или мутаций. Этот процесс будет продолжаться до тех пор, пока не будет использовано все время тестирования. Далее журналы выполнения анализируются на предмет обнаружения уязвимостей.

Также в области аудита безопасности смарт-контрактов Ethereum необходимо использовать новый инструмент *Gasper*, разработанный Тинг Ченом и другими [10]. Данный инструмент ориентирован только на выявление дорогостоящих схем программирования газа в смарт-контракте через интерфейс командной строки. Он выполняет анализ только для байт-кода, используя алгоритм Дейкстры, который вычисляет расстояние между каждым блоком, соответственно между входным и выходным. Gasper также полагается на символьное выполнение, чтобы покрыть все достижимые блоки кода, дизассемблируя его байт-код с помощью дизассемблера. В настоящее время Gasper поддерживает обнаружение трех дорогостоящих операций, включая SLOAD, SSTORE и BALANCE. Как уверяет автор статьи, в дальнейшем список операций будет пополняться [10].

*Oyente*. Oyente известен как первый и самый популярный инструмент анализа безопасности. Он был разработан генеральным директором и соучредителем Kyber Network Лой Луу [11] и является одним из немногих инструментов, представленных на крупной конференции по безопасности Ethereum Devcon. Oyente использует символическое выполнение, чтобы найти потенциальные уязвимости безопасности, включая повторный вход, зависимость от временной метки, неверно обработанные исключения. На ранних стадиях данный инструмент можно было использовать только через интерфейс командной строки. В недавних обновлениях инструмент имеет более удобный веб-интерфейс. Также стоит отметить, что это единственный инструмент, описывающий метод проверки по устранению ложных срабатываний [11].

*Securify*. Securify – это веб-инструмент для анализа безопасности смарт-контрактов. Данный инструмент обеспечивает автоматизацию проверки смарт-контрактов, гарантию обнаружения конкретных уязвимостей и расширяемость для захвата любых недавно обнаруженных уязвимостей. Securify использует формальную проверку, но также использует статический анализ кода. Securify охватывает следующие проблемы безопасности смарт-контрактов: перераспределение транзакций, рекурсивные вызовы, небезопасные шаблоны кодирования, неожиданные потоки эфира и использование ненадежных входных данных. В недавнем обновлении от 13 апреля 2020 года Securify v2.0 пополнил базу уязвимостей и поддерживает 38 новых уязвимостей, реализует новый контекстно-зависимый статический анализ и анализирует контракты, написанные на Solidity версий 0.5.8 и выше.

*SolCover*. SolCover обеспечивает покрытие кода для тестирования смарт-контрактов платформы Ethereum. Опираясь на покрытие кода, SolCover измеряет и описывает степень общего тестирования смарт-контракта. Несмотря на то, что данный метод тестирования не является механизмом для выявления конкретных уязвимостей, можно утверждать, что он создает более безопасную среду с философией, согласной которой, чем больше проведено тестов, тем выше показатель найденных уязвимостей.

**Заключение**

В результате исследования были рассмотрены основные серьезные уязвимости смарт-контрактов в блокчейне на платформе Ethereum. В данной статье были рассмотрены методы и инструменты по аудиту безопасности применительно к смарт-контрактам на платформе Ethereum. Была проведена обширная работа в поиске методов и инструментов, которые ранее не были освещены в области безопасности смарт-контрактов на платформе Ethereum.

В дальнейшем планируется дополнить список методов и инструментов по обеспечению безопасности смарт-контрактов и провести сравнительный анализ каждого из них.

**Конфликт интересов**

Не указан.

**Conflict of Interest**

None declared.

**Список литературы / References**

1. Боровик В. К вопросу о безопасности смарт-контрактов / В. Боровик и др. // Вестник Чувашского университета. – 2018. – №. 1.

2. Алиев И. А. Уязвимости смарт-контрактов блокчейн-платформы ethereum / И. А. Алиев // Научные записки молодых исследователей. – 2019. – №. 3.

3. Atzei N. A survey of attacks on Ethereum smart contracts (SoK) / N. Atzei, M. Bartoletti, T. Cimoli // Proc.of Int. Conf. on Principles of Security and Trust. Berlin, Heidelberg, Springer, 2017, pp. 164–186.

4. Iuganson A. A calculation methodology of assess for software tecurity / A. Iuganson, D. Zakoldaev. – 2017, no. 1(23), pp. 20–23.

5. Fu Y. EVMFuzzer: Detect EVM vulnerabilities via fuzz testing / Y. Fu, M. Ren, F. Ma, H. Shi, X. Yang, Y. Jiang, H. Li, and X. Shi // Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. - ESEC/FSE, Tallinn, Estonia, Aug. 2019, pp. 1110–1114.

6. Böhme M. Coverage-based greybox fuzzing as Markov chain / M. Böhme, V.-T. Pham, and A. Roychoudhury, // Proc. ACM SIGSAC Conf. Comput. Commun. Secur., Vienna, Austria, Oct. 2016, pp. 1032–1043.

7. Jiang B. Contractfuzzer: Fuzzing smart contracts for vulnerability detection / Jiang B., Liu Y., Chan W. K. //2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE). – IEEE, 2018. – P. 259-269.

8. Li N. Test oracle strategies for model-based testing / Li N., Offutt J. // IEEE Transactions on Software Engineering. – 2016. – Vol. 43. – №. 4. – P. 372-395.

9. Ashraf I. GasFuzzer: Fuzzing Ethereum Smart Contract Binaries to Expose Gas-Oriented Exception Security Vulnerabilities / Ashraf I. et al. // IEEE Access. – 2020.

10. Chen T. Under-optimized smart contracts devour your money / T. Chen, X. Li, X. Luo, and X. Zhang, // Software Analysis, Evolution and Reengineering (SANER), 2018 IEEE 24th International Conference on. IEEE, 2018, pp. 442–446.

11. Luu L. Making smart contracts smarter / Luu L. et al. // Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. – 2016. – С. 254-269.

**Список литературы на английском языке / References in English**

1. Borovik V. K voprosu o bezopasnosti smart-kontraktov [To the question of the security of smart contracts] / V. Borovik et al. //Vestnik Chuvashskogo universiteta [Bulletin of the Chuvash University]. – 2018. – №. 1. [in Russian]

2. Aliev I. A. Uyazvimosti smart-kontraktov blokchejn-platformy ethereum [Vulnerabilities of smart contracts of the ethereum blockchain platform] / I. A. Aliev // Scientific notes of young researchers [Nauchnye zapiski molodyx issledovatelej]. – 2019. – №. 3. [in Russian]

3. Atzei N. A survey of attacks on Ethereum smart contracts (SoK) / N. Atzei, M. Bartoletti, T. Cimoli // Proc.of Int. Conf. on Principles of Security and Trust. Berlin, Heidelberg, Springer, 2017, pp. 164–186.

4. Iuganson A. A calculation methodology of assess for software tecurity / A. Iuganson, D. Zakoldaev. – 2017, no. 1(23), pp. 20–23.

5. Fu Y. EVMFuzzer: Detect EVM vulnerabilities via fuzz testing / Y. Fu, M. Ren, F. Ma, H. Shi, X. Yang, Y. Jiang, H. Li, and X. Shi // Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. - ESEC/FSE, Tallinn, Estonia, Aug. 2019, pp. 1110–1114.

6. Böhme M. Coverage-based greybox fuzzing as Markov chain / M. Böhme, V.-T. Pham, and A. Roychoudhury, // Proc. ACM SIGSAC Conf. Comput. Commun. Secur., Vienna, Austria, Oct. 2016, pp. 1032–1043.

7. Jiang B. Contractfuzzer: Fuzzing smart contracts for vulnerability detection / Jiang B., Liu Y., Chan W. K. //2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE). – IEEE, 2018. – P. 259-269.

8. Li N. Test oracle strategies for model-based testing / Li N., Offutt J. // IEEE Transactions on Software Engineering. – 2016. – Vol. 43. – №. 4. – P. 372-395.

9. Ashraf I. GasFuzzer: Fuzzing Ethereum Smart Contract Binaries to Expose Gas-Oriented Exception Security Vulnerabilities / Ashraf I. et al. // IEEE Access. – 2020.

10. Chen T. Under-optimized smart contracts devour your money / T. Chen, X. Li, X. Luo, and X. Zhang, // Software Analysis, Evolution and Reengineering (SANER), 2018 IEEE 24th International Conference on. IEEE, 2018, pp. 442–446.

11. Luu L. Making smart contracts smarter / Luu L. et al. // Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. – 2016. – С. 254-269.