# Смарт-контракты: от взломов на миллиарды до безопасной эксплуатации

Тема написания и использования смарт-контрактов на территории РФ набирает обороты при развитии цифровых финансовых активов и цифровых валют. С выходом законопроекта № 1194918-8 для Федерального закона «О цифровой валюте и цифровых правах» у нас появится еще больше возможностей для использования смарт-контрактов. В этой статье рассмотрена тема безопасности таких программируемых контрактов и показано, к чему уже привели ошибки в их создании

*Текст*
**МИХАИЛ КУЛАКОВ,**
ВЕДУЩИЙ ИНЖЕНЕР-АНАЛИТИК «ДИАСОФТ»

Смарт-контракт — это решение, которое работает в блокчейн-сети и выступает «эталонным соглашением», подкрепляемым закодированным набором условий для выполнения технических и бизнес-операций. В юридическом смысле данные контракты не являются договорами. Это всего лишь приложение, запущенное для работы в блокчейн-сети и позволяющее совершать транзакции под разные специфики бизнеса, включая поставку товаров, кредитование, разного рода услуги.

Проанализировав данные о взломе и ошибках в смарт-контрактах за последние несколько лет, можно наблюдать тревожную тенденцию: за последние годы объем убытков, связанных со взломами смарт-контрактов, достиг значительных масштабов, что подчеркивает критическую важность внедрения строгих мер защиты.

**Рисунок 1. Сравнительный анализ убытков от инцидентов в смарт-контрактах, млрд долларов**

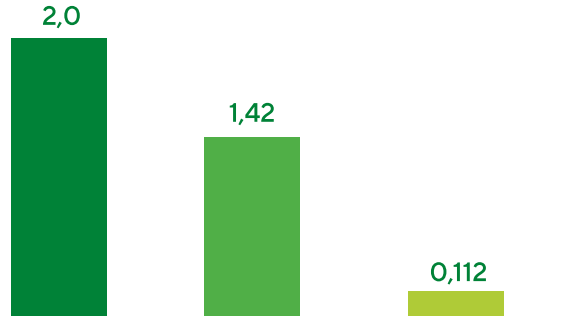


**Рисунок 2. Структура убытков по типам уязвимостей смарт-контрактов**


Источник: аналитика уязвимостей смарт контрактов, OWASP

Представленные выше данные показывают: даже если логика контракта безопасна, некорректная реализация прав доступа может привести к краху проекта. Кроме ошибок прав доступа есть и другие классы уязвимостей, например переполнение и недостаток целых чисел — эти ошибки возникают, когда арифметическая операция превышает максимальное или минимальное значение, которое может быть представлено переменной определенного типа, что приводит к «переполнению» или «оборачиванию» значения. Такие уязвимости могут позволить злоумышленнику создавать токены «из ничего» или манипулировать балансами пользователей, что напрямую приводит к финансовым потерям.

Для наглядного представления, к чему могут привести уязвимости смарт-контрактов, вспомним следующие масштабные инциденты (см. таблицу).

| Инцидент | Год | Украденные средства (оценочно), **млн долларов** | Тип уязвимости/ механизм |
|---|---|---|---|
| The DAO Hacked | 2016 | >60 | Повторный вход |
| Poly Network Hack | 2021 | 611 | Логическая ошибка моста |
| Wormhole Bridge Hack | 2022 | 325 | Логическая ошибка моста |
| Solidity 0.6.10 Exploit | 2022 | 26,5 | Переполнение целых чисел |

Таким образом, всем участникам рынка, будь то бизнес-специалисты, планирующие инвестировать в цифровые активы, или технические специалисты, пишущие код, становится очевидно, что обеспечение безопасности смарт-контрактов — это не опциональная услуга, а необходимое условие функционирования и развития всего сектора. Недостаточное внимание к этому вопросу может привести не только к частным убыткам, но и к системным рискам для всей децентрализованной экосистемы. Поэтому необходимо внедрение многоуровневой системы защиты для обеспечения безопасности смарт-контрактов, которая включает в себя комбинацию автоматизированных инструментов, ручного аудита, формальной верификации и строгого соблюдения практик разработки.

В рамках внедрения нового законодательства, связанного с использованием цифровых валют, повышается зависимость от смарт-контрактов, так как теперь они будут применяться не только для работы в приватных (корпоративных) блокчейн-сетях, таких как оператор ЦФА, но и для взаимодействия с открытыми сетями.

**Процесс аудита кода** состоит из двух компонентов: автоматизированного анализа и ручной проверки. Автоматизированные инструменты помогают быстро выявить известные шаблоны уязвимостей.

Инструменты автоматизированного контроля смарт-контрактов приведены в таблице.

| **Slither** | **Mythril** | **Echidna** |
|---|---|---|
| Инструмент статического анализа, который быстро сканирует весь код контракта на наличие распространенных уязвимостей, таких как повторный вход, ошибки управления доступом и переполнение целых чисел. Высокая скорость делает его идеальным для интеграции в CI/CD | Инструмент символьного исполнения, который пытается найти уязвимости путем анализа всех возможных путей выполнения кода, задавая условия в виде логических формул. Это позволяет находить более сложные логические ошибки, чем простой статический анализ | Инструмент фаззинга (генерации случайных тестов), который используется для проверки соблюдения свойств контракта (invariants) путем генерации большого количества случайных входных данных |

**Формальная верификация** представляет собой более строгий и математически доказуемый подход к обеспечению корректности смарт-контрактов. В отличие от тестирования, при котором проверяется работа программы на конкретных входных данных, при формальной верификации делается попытка доказать, что программа будет работать правильно для любых допустимых входных данных.

Процесс обычно включает в себя три шага:

- создание математической модели смарт-контракта и его окружения;
- формулирование требований к его поведению (спецификаций), например «баланс контракта никогда не может стать отрицательным»;
- использование специализированных инструментов для доказательства того, что модель удовлетворяет этим спецификациям.

Для этого могут применяются такие инструменты, как Certora Prover, MythX, solc-verify и другие. Верификация часто используется для наиболее критически важных частей контракта, где стоимость ошибки недопустимо высока. Главное преимущество этого метода — высокая степень уверенности в отсутствии определенного класса ошибок. В качестве примера использования верификации можно привести экосистему Cardano, в которой применяется язык программирования Plutus со встроенной поддержкой формальной верификации, что является яркой демонстрацией системного подхода к безопасности на основе математических доказательств.

Наконец, **проверенные практики и чек-листы**, которые являются фундаментом безопасной разработки и помогают избежать многих распространенных ошибок еще на этапе написания кода. К таким практикам можно отнести: использование современных компиляторов, проверенных модификаторов для контроля доступа, проверку логики управления доступом, интеграцию инструментов статического анализа в CI/CD-пайплайны и использование проверенных библиотек (например, OpenZeppelin Contracts), которые предоставляют надежную реализацию стандартов токенов и других часто используемых функций.

В совокупности аудит, верификация и соблюдение лучших практик формируют многоуровневую оборону, которая значительно снижает вероятность эксплуатации уязвимостей в смарт-контрактах. Б.О