Smart Contract UniswapV2Pair проекта

uniswap.org

Проект: uniswap.org
Сеть: Ethereum

Краткие сведения

В процессе аудита серьёзных проблем обнаружено не было, однако есть замечания.
При этом нужно понимать, что это только часть протокола UniSwap. Отсутствие серьёзных уязвимостей в этом контракте не означает их отсутствия в остальных частях протокола.

С полным списком обнаруженных замечаний можно ознакомиться ниже.

Аудит проведён для контракта UniswapV2Pair взятого из репозитория https://github.com/Uniswap/uniswap-v2-core (коммит *4dd5906*)

Обобщенные данные

Анализ контракта был произведен с помощью следующих методов:

- Статический анализ
  - Проверка кода на типичные ошибки, приводящие к наиболее распростанённым уязвимостям
- Динамический анализ
  - Запуск контракта и проведения разного рода атак с целью выявление уязвимостей
- Code Review

Полученные данные

Замечания

1. В контракте есть две функции, в которых теоретически возможна Reentrancy - уязвимость

UniswapV2Pair.swap(uint256,uint256,address,bytes)
UniswapV2Pair.burn(address)
Однако, в обоих случаях установлен модификатор lock, который реализует блокировку параллельного вызова этих методов, поэтому эти функции можно считать безопасными

2. Тоже самое касается emit’a событий после внешнего вызова.
3. Разные контракты, импортируемые в UniswapV2Pair, требуют разные версии компилятора.
4. В функции permit(address,address,uint256,uint256,uint8,bytes32,bytes32) есть проверка require(deadline >= block.timestamp, 'UniswapV2: EXPIRED'), что может быть потенциально опасно, т.к. на значение block.timestamp теоретически могут повлиять майнеры(Однако, вероятность такой атаки крайне мала).
5. В функции _update(uint balance0, uint balance1, uint112 _reserve0, uint112 _reserve1) встречается выражение uint32 timeElapsed = blockTimestamp - blockTimestampLast , где есть есть вероятность переполнения. Однако, там же есть комментарий
// overflow is desired
который говорит, что переполнение ожидаемо.
В этой же функции также используется значение block.timestamp.

Логика

1. Пользователь не взаимодействует с контрактом напрямую

Контракт служит для запуска и вызова методов другими контрактами протокола

Приложение. Классификация ошибок

Уровень

информационный	Этот вопрос не имеет прямого отношения к функциональности, но может иметь значение для понимания.
низкий	Этот вопрос не имеет никакого отношения к безопасности, но может повлиять на некоторое поведение неожиданным образом.
Средний	Проблема затрагивает некоторые функциональные возможности, но не приводит к экономически значимым потерям средств пользователей.
высокий	Эта проблема может привести к потере средств пользователя.

Вероятность

низкий	Маловероятно, что система находится в состоянии, в котором ошибка могла бы произойти или могла бы быть вызвана какой-либо стороной.
Средний	Вполне вероятно, что эта проблема может возникнуть или быть вызвана какой-либо стороной.
высокий	Весьма вероятно, что эта проблема может возникнуть или может быть использована некоторыми сторонами.

Подпись отчета

Внимание, для проверки подписи отчета необходимо выполнить следующие действия:

1. Сделать резервную копию отчета
2. Преобразовать (а лучше изначально иметь) в версию docx.
3. Удалить данную страницу, так как она не участвует в подписи
4. Получить хеш-сумму sha256 от файла данного отчета.
5. Заполнить данную форму, где
   step1: Ethereum адрес с сайта telescr.in,
   step2: подпись, указанную ниже (поле msg)
   step3: хеш, полученный в пункте 4.

Аналогичный вариант проверки:
Внести данную подпись на сайте https://telescr.in/verify

Sha265 docx файла:
10721caf6d55ae9574b8c31bca68fc8ba6b1795a194656007fc2378890166efd
Подпись:


```
{
  "address": "0x505ade8cea4db608250e503a5e8d4cb436044d2e",
  "msg":
  "10721caf6d55ae9574b8c31bca68fc8ba6b1795a194656007fc2378890166efd",
  "sig":
  "0xa3ff3f89f107da45eb33bdfb4e144e32c8e1bd33bdc6d50ea6b6f898b92404834aeb0
  ba9f23aca90acc398750918ae842c344c96e0036d8cb595edf8b56ab71b1b",
  "version": "3"
}
```