CHANGED 2 MINUTES AGO

Final Security Audit of `DexioLock` contract.

Conclusion

This audit was made by Web3Go
Auditor: Vladimir Smelov vsmelov.job@gmail.com.
Date: 2022-08-09

The following items were **NOT FOUND** in:

- Backdoors for investor funds withdrawal by anyone.
- Bugs that allow stealing money from the contract.
- Any severe security problems.
- Any serious problems with gas consumption.

The client was acknowledged about all notes below.

Scope

Pre-audit

- DexioLock.sol (md5 hash - `f948096a0ae089f57977fdc1fe47fdcc`)

Final audit

- DexioLock.sol
  https://gist.github.com/vsmelov/7d3a26e56534049990d37d620a59b0f7

Methodology

1. Blind audit. Understand the structure of the code without reading any docs.
2. Ask questions to developers.
3. Draw the scheme of cross-contracts interactions.
4. Write user stories and usage cases.
5. Run static analyzers.
6. Find problems with:

- backdoors;
- bugs;
- math;
- potential leaking of funds;
- potential locking of the contract;
- validate arguments and events;
- others.

Result

Critical

There are no critical issues in the final version of the contract.

Major

There are no major issues in the final version of the contract.

Warning

WARNING-1. Transfer method inconsistency.

At:

- contracts/DexioLock.sol:545

In unlock you use

```
545 IERC20(userLock.token).safeTransfer(msg.sender,   unlockAmount);      //z
```

But at

- contracts/DexioLock.sol:481

You use

```
481 safeTransferFromEnsureExactAmount(token,   msg.sender,   address(this),
```

why don't you use the same functions for lock and unlock?

Recommendation.

Consider the usage of the same functions to lock/unlock tokens or adding comments or refactor.

Status.

ACKNOWLEDGED - **not a security issue**.

Client's commentary.

> It's to save gas; if someone was able to put tokens via safeTransferFromEnsureExactAmount then it means that token is NORMAL (actual transfer amount equals to passed amount). So you can let the user withdraw it back in a usual way.

Comment.

COMMENT-1. Redundant struct.

At

- contracts/DexioLock.sol:422

```
422 struct  CumulativeLo ckInfo  {
423     address  token;
424     uint256  amount;
425 }
```

The token is a key to the mapping
The factory is never set.
The amount is never used for any operational purposes (only in view method).

Recommendation.

Consider removing this struct.
Or, at least, create a direct

```
mapping(address => uint256) public cumulativeLockInfo;
```

Or use just

```
IERC20(token).balanceOf(address(this));
```

to check how much tokens are locked in the contract.

Status.

PARTIALLY FIXED - `factory was removed from the struct`.

COMMENT-2. Use mappings rather than arrays.

At

- contracts/DexioLock.sol:430

```
430 Lock[] private _locks;
```

Mappings are cheaper.

Iteration over array is never used inside transactional (not-view) methods.

Also, you will be able remove the item and receive gas compensation.

Recommendation.

Consider using a mapping.

Status.

ACKNOWLEDGED

COMMENT-3. Bad code style.

At

- contracts/DexioLock.sol:431

```
431     mapping(address => EnumerableSet.UintSet)
432     private _userNormalLockIds;
```

- contracts/DexioLock.sol:590

```
590     CumulativeLockInfo storage tokenInfo = cumulativeLockInfo[
591     userLock.token
592     ];
```

The identation is counter-intuitive.

Recommendation.

Follow the identation code-style from https://docs.soliditylang.org/en/v0.8.16/style-guide.html

Status.

ACKNOWLEDGED

COMMENT-4. Confusing the “NormalToken” suffix.

At

- contracts/DexioLock.sol

You use the suffix “Normal” often, but it’s unclear what you mean by “normal.”

Recommendation.

Consider adding a definition of the “normal token” as a comment.

Status.

ACKNOWLEDGED

COMMENT-5. Redundant struct.

At

- contracts/DexioLock.sol:436

```
436 mapping(address => EnumerableSet.UintSet) private _tokenToLockIds;
```

It is not needed. The frontend may fetch events. Just a wasting of gas.

Recommendation.

Consider removing or adding comments on why it is needed.

Status.

ACKNOWLEDGED

COMMENT-6. Unused event.

At

- contracts/DexioLock.sol:437

```
437 event Log(uint gas);
```

The event is never used.

Recommendation.

Remove.

Status.

FIXED - The event was removed.

COMMENT-7. Redundant check.

At

- contracts/DexioLock.sol:461

```
460 modifier validLock(uint256 lockId) {
461     require(lockId < _locks.length, "Invalid lock id");
462     _;
463 }
```

Accessing an array past its end causes a failing assertion. Methods .push() and .push(value) can be used to append a new element at the end of the array, where .push() appends a zero-initialized element and returns a reference to it.

Check this - https://docs.soliditylang.org/en/v0.6.12/types.html

Recommendation.

Remove.

Status.

ACKNOWLEDGED

COMMENT-8. Use indentation.

At

- contracts/DexioLock.sol:509-516

```
509         Lock memory newLock = Lock({
510         id: id,
511         token: token,
512         owner: owner,
513         amount: amount,
514         lockDate: block.timestamp,
515         unlockDate: unlockDate
516         });
```

The identations is wrong. Follow the solidity style-guide.

Recommendation.

Refactor

Status.

ACKNOWLEDGED

COMMENT-9. Rephrase the error message.

At

- contracts/DexioLock.sol:523

Recommendation.

Refactor

Status.

ACKNOWLEDGED

COMMENT-10. Additional timestamp sanity check.

At

- contracts/DexioLock.sol:476

```
475  require(
476      unlockDate > block.timestamp,
477      "Unlock date should be after current time"
478  );
```

Sometimes frontend may mess up with seconds and microseconds (x1000 bigger), so it's wise to check, `require(unlockDate - block.timestamp < 1000 days)`

Recommendation.

Consider adding additional check on `unlockDate`.

Status.

ACKNOWLEDGED

COMMENT-11. Function duplication.

At

- contracts/DexioLock.sol:664
- contracts/DexioLock.sol:633

```
662 function totalTokenLockedCount() external view returns (uint256)  {
663     return allNormalTokenLockedCount();
664 }
```

Two functions do the same work.

Recommendation.

Consider removing one of two identical functions.

Status.

ACKNOWLEDGED

COMMENT-12. Impossible condition.

At:

- contracts/DexioLock.sol:529

```
528 uint256 unlockAmount = userLock.amount;
529 if (IERC20(userLock.token).balanceOf(address(this)) < unlockAmount)
530     unlockAmount = IERC20(userLock.token).balanceOf(address(this));
531 }
```

It’s not clear in which use case it’s possible.

Recommendation.

Consider adding comments or example of the usage scenario when it is possible.

Status.

ACKNOWLEDGED