Security Audit of Galaxe

Conclusion

In the final contract were not found:

- Backdoors for investor funds withdrawal by anyone.
- Bugs allowing to steal money from the contract.
- Other security problems.

Obvious errors or backdoors were not found in the contract.
The client was acknowledged about all secutiry notes below.

SECURED 100% SECURED

Scope

first audit scope:
1ca77020962c6c31d51b93d849ef0af8 Archive.zip

```
$ tree contracts
contracts
├── CollectionERC1155.sol
├── CollectionERC721.sol
├── Market.sol
├── interfaces
│   ├── ICollectionERC1155.sol
│   └── ICollectionERC721.sol
├── lib
│   ├── LibConvert.sol
│   ├── LibListing.sol
│   ├── LibSig.sol
│   ├── LibToken.sol
│   └── SafeERC20.sol
└── test
└── MintableToken.sol
```

second audit scope:
https://github.com/poegva/galaxe/tree/main/contracts
(https://github.com/poegva/galaxe/tree/main/contracts)
commit: 6929b70

Methodology

1. Blind audit. Try to understand the structure of the code.
2. Find info in internet.
3. Ask quiestions to developers.
4. Draw the scheme of cross-contracts interactions.
5. Write user-stories, usage cases.
6. Run static analyzers

Find problems with:

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

Result

Critical

Not found.

Major

✅ 1. Use safeERC20.

At:

- contracts/Market.sol:309
- contracts/Market.sol:313
- contracts/Market.sol:158
- contracts/Market.sol:160
- contracts/Market.sol:227

you ignore success-status from IERC20.

Recommendation

Use safeERC20 methods.

Status

FIXED

Warning

🆗 1. Royalty for ERC1155.

At:

- contracts/lib/LibToken.sol:54

the royalty is processed only for erc721.
It's better to add support for royalty for erc1155 or use exact `require` at

- contracts/lib/LibToken.sol:54

or use specific type check at

- contracts/Market.sol:306

Recommendation

Add royalty for erc1155 or be exact with getRoyalty type-checks.

Client's comment

> Royalties only for ERC721 are by-design

Status

ACKNOWLEDGED

🆗 2. Potential reentry.

At:

- contracts/Market.sol:101
- contracts/Market.sol:136
- contracts/Market.sol:161
- contracts/Market.sol:186
- contracts/Market.sol:204
- contracts/Market.sol:229
- contracts/Market.sol:242
- contracts/Market.sol:249

you change the state after external calls.
It's a bad habbit since it may lead to reentry attack.

Recommendation

Change state before external calls.
Use nonReentrant modifier.

Client's comment

> Trusted contracts are interacted, so CEI pattern is not neccesary

Status

ACKNOWLEDGED

🆗 3. Unfair cancelation of started auction.

At:
- contracts/Market.sol:227

it's not fair that auctioneer may cancel auction after some bid was placed.
Bidder already payed gas.

Recommendation

Remove it or add warning comment in the code.

Client's comment

> Such possibility is by-design

Status

ACKNOWLEDGED

Comment

🆗 1. Use Openzeppelin imports instead of implementing contracts and libs by yourself.

It seems like several contracts code were copy-pasted from Openzeppelin github.
e.g.
- SafeERC20.sol
- LibConvert.sol
  It's better to use imports from well-known well-tested repository. It decrease the number of files in the repo and decrease the chance of typo mistake. Or if some

modification is neccesary, add reference link in a comment to the original.

Recommendation

Use imports or add references.

Status

ACKNOWLEDGED

🆗 2. LibConvert is not crystal cearl.

The code in

- LibConvert.sol
  is not crystal clear. Can your provide more comments and references about how does it work?

Recommendation

Add more info.

Client’s comment

> LibConvert is used to convert base64 to base58 encoding in order to store IPFS hashes as single bytes32

Status

ACKNOWLEDGED

✅ 3. Freeze solc version.

Everywhere you use
pragma solidity ^0.8.9;
it better to replace it with exact
pragma solidity 0.8.9;
The main reason for it, is that you want empty external auditor company and everyone wants to mark it as an issue (but basically it’s not). Look at the audits of CERTIK and SolidProof, they will mark it as a big red cross :-(

Recommendation

Use
pragma solidity 0.8.9;

Status

FIXED

✅ 4. Add zero-check.

At:

- contracts/Market.sol:64
- contracts/Market.sol:65

you don't have check for zero-address.
This is kind of security standard.
See **https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation**

(https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation)

Recommendation

Use
require(value != address(0), "zero address");

Status

FIXED

✅ 5. Pack struct efficiently.

At:

- contracts/lib/LibListing.sol:39

it's better to place address field the last, to auto-combine it together with two bool, and use much less storage space.
See - **https://medium.com/@novablitz/storing-structs-is-costing-you-gas-774da988895e**

(https://medium.com/@novablitz/storing-structs-is-costing-you-gas-774da988895e)

Recommendation

Re-struct.

Status

FIXED

🆗 6. Rename royalty to royaltyNumerator.

In fact royalty is calculated on 10000 denominator, so it's better to use royaltyNumerator as a var name to avoid misusing.

Recommendation

Rename.

Client's comment

> LibConvert is used to convert base64 to base58 encoding in order to store IPFS hashes as single bytes32

Status

ACKNOWLEDGED

🆗 7. Exclude seller from createListing argument.

At:
- contracts/Market.sol:93

you can exclude this field from the arg struct.

Recommendation

Optimize arg struct.

Client's comment

> This will require creating spearate structure and combining several structures accross code, decreasing clarity without significant gas savings; decided to remain unchanged

Status

ACKNOWLEDGED

🆗 8. Add limit on listing duration.

At:
- contracts/Market.sol:94

you can add check for listing duration to avoid misusing.

Recommendation

Add
require(listing.end - listing.start > '365 days', "too long");
instead of
require(listing.end > listing.start, "Market: end later than start");

Client's comment

> This is business logic change, not safety-related

Status

ACKNOWLEDGED

🆗 9. Unclear 0x1901.

At:

- contracts/Market.sol:120
- contracts/CollectionERC721.sol:96
- contracts/CollectionERC1155.sol:86
  you use 0x1901, but it’s not clear what is it.

Recommendation

Add more info.

Client’s comment

> it’s EIP712 prefix

Status

ACKNOWLEDGED

✅ 10. Use `external` modifier.

At:

- contracts/CollectionERC1155.sol:56
- contracts/CollectionERC1155.sol:95
- contracts/CollectionERC721.sol:147
- contracts/CollectionERC721.sol:160
- contracts/CollectionERC1155.sol:130
- contracts/CollectionERC1155.sol:150

you can use `external` modifier to save gas on calls, since in this case EVM will read arguments from calldata directly.

Recommendation

Use `external` modifier, save gas.

Client’s comment

> Changed where possible, but domainHash is used inside contract; also inherited methods visibility can’t be changed

Status

FIXED

🆗 11. DEFINE CHAINID AS IMMUTABLE VARIABLE.

At:

- contracts/CollectionERC721.sol:106
- contracts/CollectionERC1155.sol:96

you copy chainId to the stack every time.
It may save some gas to initialize chainId in constructor as an `immutable` variable, then it will be included to the opcode.
But i'm not sure, some tests are required.

Recommendation

Use `immutable` chainId, save gas.

Client's comment

> Chain ID can theoretically be altered; also chainId opcode costs only G_base

Status

ACKNOWLEDGED

🆗 12. NAME INTERNAL METHODS STARTING WITH `_` .

At:

- contracts/lib/LibToken.sol:61

Recommendation

Rename to `_transfer` .

Client's comment

> It's library, not contract, so internal methods should not start with _

Status

ACKNOWLEDGED

✅ 13. USE SAFE METHODS FOR ERC721.

At:

- contracts/lib/LibToken.sol:64
- contracts/lib/LibToken.sol:88

it's more advanced to use safe method.

Recommendation

Use `IERC721.safeTransferFrom`.

Status

FIXED

🆗 14. Incorrect event names.

At:
- contracts/Market.sol:145

Payment and token were not really claimed.

Recommendation

Use others names.

Client's comment

> Decided to remain unchanged in order to preserve compatibility

Status

ACKNOWLEDGED

✅ 15. Set token properties before `_mint`.

At:
- contracts/CollectionERC1155.sol:124

you first do mint and then set properties.
However there is a onERC1155Received inside `_mint`, so if receiver will try to process the token properties in some way this will fail.

Recommendation

Set token properties before `_mint`.

Status

FIXED

✅ 16. Exact check for listingType.

At:
- contracts/Market.sol:147

it's better to do exact check, what listingType do you expect.

Recommendation

Use exact check for listingType.

Status

FIXED

🆗 17. Use standard implementation for `isApprovedForAll`.

At:

- contracts/CollectionERC721.sol:167

instead of overriding the method and using more gas you can set approveForAll for the operator=market and use standard method implementation.

Recommendation

Use standard implementation for `isApprovedForAll`.

Client's comment

> This logic is used to make market approved for all tokens in collection, undifferent of owner, in order that users don't spend gas pre-approving market contract for marketplace's internal collections. There is no default implementation for that.

Status

ACKNOWLEDGED

✅ 18. Emit event on new contracts created.

At:

- contracts/Market.sol:242
- contracts/Market.sol:249

it makes sense to emit event, otherwise it will be difficult to analyze the history.

Recommendation

Emit events.

Status

FIXED

✅ 19. The lastBid set is not need.

At:

- contracts/Market.sol:102

This is not really need, since you check it here

- contracts/Market.sol:150

Recommendation

Remove the set.

Status

FIXED

✅ 20. Increase the bid for the same bidder.

At:

- contracts/Market.sol:158

if someone wants to increase his own bid from 1000 to 1200, there is no need to transfer 1000 back to bidder and then 1200 to the contract.
You can just send 200 more tokens to the contract.

Recommendation

Add

```
if (details.lastBidder == msg.sender) {
    currency.transfer(details.lastBidder, amount - details.lastBid);
}
```

or add another `increaseBid` method for such action.

Status

FIXED

✅ 21. Emit event on royalty and payment paid.

At:

- contracts/Market.sol:309
- contracts/Market.sol:313

it’s better to emit event for royalty and payment paid for easy data analysis.

Recommendation

Emit event.

Status

FIXED

Slither static-analyzer log

https://pastebin.com/yi3gYfTn (https://pastebin.com/yi3gYfTn)