



Smart contracts security assessment

Final report

[Tariff: Standard](#)

Elite Protocol

September 2023



0xguard.com



hello@0xguard.com

Contents

1. Introduction	3
2. Contracts checked	5
3. Procedure	5
4. Known vulnerabilities checked	6
5. Classification of issue severity	7
6. Issues	7
7. Conclusion	13
8. Disclaimer	14

Introduction

The report has been prepared for **Elite Protocol**.

The reviewed project is a Tomb Finance fork, allowing users to farm the main Elite (ELITE) and the share Elite Share (ESHARE) tokens. ELITE and ESHARE tokens are ERC20-standard tokens with taxes on transfers.

Both rewards pool (a.k.a. farms) contracts may charge a fee of up to 5% for each deposit.

The SHA-1 hashes of audited files are:

Treasury.sol 41432030e309437b4cadd5d3f1cdf1cd4158d233

Bond.sol 16ee22b1173867350415ccc414708106e99e9f4d

ShareRewardPool.sol b227386a19554ca457b2eb27152c55f05027369f

Boardroom.sol 7113a08cfd849a0302a208ddceff134112e16794

Oracle.sol 80108a041ae9136806c604857209213540310cb5

TokenDistributor.sol 9deaf7f73bb615526c3295fccff76cee4608543

Test.sol ca4b7b899b5ab2f2e5f99349e857aaf00ef46a4e

GenesisRewardPool.sol 47c57d9e46579bbd5440747c76fe8b4e86eb93be

ITreasury.sol 1d7b352a5c2a2d5c217a46f85aa22a0ea87c0870

contractguard.sol bca70f79eb2cba037e9cfbf995f8c83a6e6e95b8

lBoardroom.sol e4eb741c202150a6d13df1ed9fa5b7acb8d67338

Babylonian.sol 2ed3a1bd33b91eb85840039f2fe17a8ddfa04708

IBasisAsset.sol 487cac6a4b738ecad2e7cfb337a2b9b68af0bd78

ShareWrapper.sol 90261cd37e5609717d8a6ab1bc723e1880dc0666

Operator.sol 43622b005ccc6380d1077ffe55d369bbd188e001

SafeMath8.sol b354f3ba901d9867b7773895fa7202cb3d531f6c

IUniswapV2Pair.sol 859621275b0e8e832175423cf25f46cfb50d227f

Share.sol cc9baa5bcc6c67dddc793de3f1bd4c57335cd662

IOracle.sol cf1523d8a325ddf45a54b72194a7b467a174b54c

IUniswapV2Router02.sol ca5af330b407d431797999717eae83326fba6616

IUniswapV2Router01.sol 41e79e97676c8701e825b570e32a3010ab542bbc

Update. The updated code was deployed to the following addresses in the Base Chain:

Token Distributor : [0x431D0cd1Dd8A2CeD36dd8717057BA2B332fE9360](https://base.etherscan.io/address/0x431D0cd1Dd8A2CeD36dd8717057BA2B332fE9360)

BoardRoom : [0xD07EC7e87a67bdCA84255b223C72fb9a6190C10f](https://base.etherscan.io/address/0xD07EC7e87a67bdCA84255b223C72fb9a6190C10f)

Treasury : [0xd0d1C9CE5556f1e48651977F74E0f7BA26Fdedd1](https://base.etherscan.io/address/0xd0d1C9CE5556f1e48651977F74E0f7BA26Fdedd1)

GenesisRewardPool : [0xc61b13e285981A9E17943E4dCAf41e50635F23D4](https://base.etherscan.io/address/0xc61b13e285981A9E17943E4dCAf41e50635F23D4)

ShareRewardPool : [0x5A6c5819B4BbAAD6f6122c21c979120663a58F6c](https://base.etherscan.io/address/0x5A6c5819B4BbAAD6f6122c21c979120663a58F6c)

Bond : [0x57bD28bC3EeAc0dBbAB5bA5487d454d30d81583c](https://base.etherscan.io/address/0x57bD28bC3EeAc0dBbAB5bA5487d454d30d81583c)

Native : [0x33994a876c342Aa8bd25F70961810a7E103218B0](https://base.etherscan.io/address/0x33994a876c342Aa8bd25F70961810a7E103218B0)

ShareToken : [0x5e8a9f92ae2a341DC9062100182493085F6C490f](https://base.etherscan.io/address/0x5e8a9f92ae2a341DC9062100182493085F6C490f)

Oracle : [0x6631FA817614CF2d8e395E10C7bb0e660c2A891F](#)

Name	Elite Protocol
Audit date	2023-09-05 - 2023-09-06
Language	Solidity
Platform	Base Chain

Contracts checked

Name	Address
Elite (formerly Test)	0x11e14Cc475f9Da7E82ba8Ca3842350EaA33453c2
Share	0x6d0EFc8698594B135C26727C202b55459D511E2D
Boardroom	0x95929958E4B8a72D1Da459EAFA2600053C977d75
Bond	0x5e26625d0228BE64f0A131A4f7CA0E2167d01d58
Treasury	0x50B94868c649D4F44CFf35d6cb44dC3630911E86
GenesisRewardPool	0x92aa11EDF6b9202e6F258f1feed9673A1A3Cec33
ShareRewardPool	0xBe20eaBF2F861d328B5995C4750c1Fc34B1831aA
Oracle	0x75d5b4d02C5B4888988b092179198a72db415a7d

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) of all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
<u>Unencrypted Private Data On-Chain</u>	passed
<u>Code With No Effects</u>	passed
<u>Message call with hardcoded gas amount</u>	passed
<u>Typographical Error</u>	passed
<u>DoS With Block Gas Limit</u>	passed
<u>Presence of unused variables</u>	passed
<u>Incorrect Inheritance Order</u>	passed
<u>Requirement Violation</u>	passed
<u>Weak Sources of Randomness from Chain Attributes</u>	passed
<u>Shadowing State Variables</u>	passed
<u>Incorrect Constructor Name</u>	passed
<u>Block values as a proxy for time</u>	passed
<u>Authorization through tx.origin</u>	passed
<u>DoS with Failed Call</u>	passed
<u>Delegatecall to Untrusted Callee</u>	passed
<u>Use of Deprecated Solidity Functions</u>	passed
<u>Assert Violation</u>	passed
<u>State Variable Default Visibility</u>	passed
<u>Reentrancy</u>	passed

<u>Unprotected SELFDESTRUCT Instruction</u>	passed
<u>Unprotected Ether Withdrawal</u>	passed
<u>Unchecked Call Return Value</u>	passed
<u>Floating Pragma</u>	passed
<u>Outdated Compiler Version</u>	passed
<u>Integer Overflow and Underflow</u>	passed
<u>Function Default Visibility</u>	passed

Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

No issues were found

Medium severity issues

1. Possibly unlimited deposit fee (ShareRewardPool)

Status: Fixed

The owner can add new pools with deposit fee not greater than 1%, but the `set` function has no such safety checks, allowing owner to stop all deposits by setting deposit fee over 100%.

Recommendation: Include safety check similar to the one in `add` function.

Low severity issues

1. Lack of events (Elite (formerly Test))

Status: Open

There's a general lack of events emitted in governance functions, which complicates tracking the history of changes in crucial system parameters.

2. Public open function (Elite (formerly Test))

Status: Open

The `setPairDshare()` initializer function is open for public use. While the deployed contract is initialized correctly, any possible future code reuse may face difficulties.

3. Gas optimization (Elite (formerly Test))

Status: Open

1. There're multiple unnecessary readings of data from blockchain in the `start()` function: `BUSD.balanceOf(address(this))` and `balanceOf(address(this))` should be read once to local variables.
2. The `mint()` function contains a double call of the contract's balance before and after with an unclear purpose. The internal `_mint()` function inherited from the ERC20 contract has its own safety checks.
3. Unnecessary multiplication by `MULTIPLIER` is performed in the `transfer()` and

`transferFrom()` functions. It doesn't improve the accuracy of calculations, the divisor should be set to 100 instead.

4. Unused code (Elite (formerly Test))

Status: Open

The SafeMath8 library is not in use. It's also outdated since it should be compiled with pre-0.8 pragma versions to avoid double-spending gas on overflow checks.

5. Parameters of addLiquidity (Elite (formerly Test))

Status: Open

There's a `addLiquidity` call for a Uniswap-like router in the `start()` function that may constantly fail since the pair is already created and the amounts are fixed. Also, the deadline parameter is used incorrectly since it can't be calculated on-chain: router contract checks deadline is not smaller than the current time, i.e. setting a deadline to `block.timestamp` or greater will always pass and setting it lower than `block.timestamp` will always revert.

6. Lack of events (Share)

Status: Open

There's a general lack of events emitted in governance functions, which complicates tracking the history of changes in crucial system parameters.

Recommendation: We recommend reviewing the architecture for blocking users when selling tokens. Perhaps the `_transfer()` function should track token **senders** instead of **recipients**.

7. Gas optimization (Share)

Status: Open

1. Unnecessary multiplication by `MULTIPLIER` is performed in the `transfer()` and `transferFrom()` functions. It doesn't improve the accuracy of calculations, the divisor should be set to 100 instead.

8. ContractGuard doesn't prevent re-entrancy (Boardroom)

Status: Open

The ContractGuard contract is designed to prevent multiple calls in the same block but it doesn't prevent re-entrancy, i.e. multiple calls in the same transaction.

```
modifier onlyOneBlock() {
    require(!checkSameOriginReentranted(), "ContractGuard: one block, one
function");
    require(!checkSameSenderReentranted(), "ContractGuard: one block, one
function");

    _;

    _status[block.number][tx.origin] = true;
    _status[block.number][msg.sender] = true;
}
```

9. Validation in the initialize() function (Treasury)

Status: Open

The input parameters of the `initialize()` function aren't checked in any way. The `nativePriceOne` variable is set to `1e18` regardless of the actual `decimals()` value of the native token.

10. ContractGuard doesn't prevent re-entrancy (Treasury)

Status: Open

The ContractGuard contract is designed to prevent multiple calls in the same block but it doesn't prevent re-entrancy, i.e. multiple calls in the same transaction.

```
modifier onlyOneBlock() {
    require(!checkSameOriginReentranted(), "ContractGuard: one block, one
function");
    require(!checkSameSenderReentranted(), "ContractGuard: one block, one
function");

    _;

    _status[block.number][tx.origin] = true;
    _status[block.number][msg.sender] = true;
}
```

11. Gas optimization (Treasury)

Status: Open

1. Checking the `targetPrice` from parameters of the `redeemBonds()` function seems unnecessary: the gas-wise way is to receive a `bondRate` parameter and check it against `getBondPremiumRate()` directly.
2. Excessive data is read from the blockchain in the `redeemBonds()` function: `getBondPremiumRate()` should receive already in-memory values of `nativePrice` and `nativePriceCeiling`.
3. Redundant code in the `redeemBonds()` function: `require(_rate > 0)` is always passed as it's already checked that `nativePrice > nativePriceCeiling`.

12. Possible block gas limit problem (GenesisRewardPool)

Status: Open

An unlimited loop over an array of pools may cause a gas limit problem if too many pools would be added. The owner must pay attention when adding new pools.

13. Gas optimization (GenesisRewardPool)

Status: Open

Pool duplication check is ineffective, it should be performed via mapping from the token address. The other way is to allow duplicated pools by storing individual pools balances in `PoolInfo` structure, i.e. the `updatePool()` function should not check the `pool.token.balanceOf(address(this))` but read the pool balance from the structure.

14. Contract doesn't support tokens with transfer fees (GenesisRewardPool)

Status: Open

Actual transfer amounts aren't checked so the owner must not add pools with tokens with transfer commissions unless this contract is excluded from such fees (see Test and Share contracts).

15. Contract doesn't support tokens with transfer fees (ShareRewardPool)

Status: Open

Actual transfer amounts aren't checked so the owner must not add pools with tokens with transfer

commissions unless this contract is excluded from such fees (see Test and Share contracts).

16. Gas optimization (ShareRewardPool)

Status: Open

Pool duplication check is ineffective, it should be performed via mapping from the token address. The other way is to allow duplicated pools by storing individual pools balances in `PoolInfo` structure, i.e. the `updatePool()` function should not check the `pool.token.balanceOf(address(this))` but read the pool balance from the structure.

17. Possible block gas limit problem (ShareRewardPool)

Status: Open

An unlimited loop over an array of pools may cause a gas limit problem if too many pools are added. The owner must pay attention when adding new pools.

18. Gas optimization (Oracle)

Status: Open

The `getCurrentEpoch()` function and `epoch` variable of the Epoch contract have unclear functionality.

Conclusion

Elite Protocol Elite (formerly Test), Share, Boardroom, Bond, Treasury, GenesisRewardPool, ShareRewardPool, Oracle contracts were audited. 1 medium, 18 low severity issues were found. 1 medium severity issue has been fixed in the update.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without OxGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts OxGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

OxGuard retains exclusive publishing rights for the results of this audit on its website and social networks.

