OBELISK

Part of Tibereum Group

AUDITING REPORT

Version Notes

Version	No. Pages	Date	Revised By	Notes
1.0	Total: 22	2022-01-28	Zapmore, @Plemonade	Audit Final

Audit Notes

Audit Date	YYYY-MM-DD - YYYY-MM-DD
Auditor/Auditors	Plemonade, ByFixter
Auditor/Auditors Contact Information	contact@obeliskauditing.com
Notes	Specified code and contracts are audited for security flaws. UI/UX (website), logic, team, and tokenomics are not audited.
Audit Report Number	OB565456222

Disclaimer

This audit is not financial, investment, or any other kind of advice and is for informational purposes only. This report is not a substitute for doing your own research and due diligence. Obelisk is not responsible or liable for any loss, damage, or otherwise caused by reliance on this report for any purpose. Obelisk has based this audit report solely on the information provided by the audited party and on facts that existed before or during the audit being conducted. Obelisk is not responsible for any outcome, including changes done to the contract/contracts after the audit was published. This audit is fully objective and only discerns what the contract is saying without adding any opinion to it. The audit is paid by the project but neither the auditors nor Obelisk has any other connection to the project and has no obligations other than to publish an objective report. Obelisk will always publish its findings regardless of the outcome of the findings. The audit only covers the subject areas detailed in this report and unless specifically stated, nothing else has been audited. Obelisk assumes that the provided information and material were not altered, suppressed, or misleading. This report is published by Obelisk, and Obelisk has sole ownership of this report. Use of this report for any reason other than for informational purposes on the subjects reviewed in this report including the use of any part of this report is prohibited without the express written consent of Obelisk. In instances where an auditor or team member has a personal connection with the audited project, that auditor or team member will be excluded from viewing or impacting any internal communication regarding the specific audit.

Obelisk Auditing

Defi is a relatively new concept but has seen exponential growth to a point where there is a multitude of new projects created every day. In a fast-paced world like this, there will also be an enormous amount of scams. The scams have become so elaborate that it's hard for the common investor to trust a project, even though it could be legit. We saw a need for creating high-quality audits at a fast phase to keep up with the constantly expanding market. With the Obelisk stamp of approval, a legitimate project can easily grow its user base exponentially in a world where trust means everything. Obelisk Auditing consists of a group of security experts that specialize in security and structural operations, with previous work experience from among other things, PricewaterhouseCoopers. All our audits will always be conducted by at least two independent auditors for maximum security and professionalism.

As a comprehensive security firm, Obelisk provides all kinds of audits and project assistance.

Audit Information

The auditors always conducted a manual visual inspection of the code to find security flaws that automatic tests would not find. Comprehensive tests are also conducted in a specific test environment that utilizes exact copies of the published contract.

While conducting the audit, the Obelisk security team uses best practices to ensure that the reviewed contracts are thoroughly examined against all angles of attack. This is done by evaluating the codebase and whether it gives rise to significant risks. During the audit, Obelisk assesses the risks and assigns a risk level to each section together with an explanatory comment. Take note that the comments from the project team are their opinion and not the opinion of Obelisk.

Table of Contents

Project Information

Name	SecureDAO
Description	SecureDAO is a community-driven and security-focused protocol that helps you earn passively on your Crypto portfolio. SecureDAO uses innovative treasury management and protocol-owned liquidity to generate shared profits with holders of SCR.
Website	https://www.securedao.finance/
Contact	rabage#0897 on Discord
Contact information	rabage#0897 on Discord
Token Name(s)	Secure and Staked Secure
Token Short	SCR and sSCR
Contract(s)	See Appendix A
Code Language	Solidity
Chain	FTM

Audit of SecureDao

All issues found were either closed or mitigated by the project team.

Obelisk was commissioned by SecureDAO on the 10th of January 2022 to conduct a comprehensive audit of SecureDAOs' contracts. The following audit was conducted between the 10th of January 2022 and the 27th of January 2022. Two of Obelisk's security experts went through the related contracts manually using industry standards to find if any vulnerabilities could be exploited either by the project team or users.

The auditors found 5 findings of Low and Medium risk. These were all resolved in some way by the project team. Issue #01, #08, and #09 all are only mitigated instead of closed since the project team has implemented timelocks on the advised contracts. However, these timelocks are only 12 hours long instead of the recommended 72 hours.

The informational findings are good to know while interacting with the project but don't directly damage the project in its current state, hence it's up to the project team if they deem that it's worth solving these issues.

The team has not reviewed the UI/UX, logic, team, or tokenomics of the SecureDAO project**.**

Please read the full document for a complete understanding of the audit.

Summary Table

Finding	ID	Severity	Status
No Timelock	#0001	Low Risk	Mitigated
UI May Modify Depositor Address	#0002	Medium Risk	Closed
Users Redeeming Can Be Blocked Indefinitely	#0003	Low RIsk	Closed
High Upper Bound On Fees	#0004	Informational	Closed
No Events Emitted For Changes To Protocol Values	#0005	Informational	Closed
Missing Variable In Emit	#0006	Informational	Closed
Contract Name Not Matching Filename	#0007	Informational	Closed
No Timelock For The Price Feed	#0008	Medium Risk	Mitigated
Low Minimum Timelock Delay	#0009	Low RIsk	Mitigated

Findings

Manual Analysis

No Timelock

FINDING ID	#0001
SEVERITY	Low Risk
STATUS	Mitigated
LOCATION	• BondDepositoryV2.sol -> 154-173: *function setBondTerms (PARAMETER parameter_, uint input_) external onlyOwner();* • BondDepositoryV2.sol -> 182-198: *function setAdjustment (bool addition_,uint increment_,uint target_,uint32 buffer_) external onlyOwner();* • BondDepositoryV2.sol -> 204-207: *function setStaking(address staking_) external onlyOwner();* • BondDepositoryV2.sol -> 213-216: *function setStakingHelper(address stakingHelper_) external onlyOwner();*

DESCRIPTION	Important functions should be limited to a contract under the ownership of a timelock. No timelock has been provided to Obelisk.
RECOMMENDATION	Obelisk recommends a minimum delay of 72 hours.
RESOLUTION	A timelock was added for both SecureBondDepositoryV2 contract and ChainlinkBondingCalculator. Note: timelock delay is 12 hours. Timelock 0xac4220abfd028f9c12b7916235180bbe73619b00

UI May Modify Depositor Address

FINDING ID	#0002
SEVERITY	Medium Risk
STATUS	Closed
LOCATION	BondDepositoryV2.sol -> 230-286

```
function deposit(
    uint amount_,
    uint maxPrice_,
    address depositor_
) external returns ( uint ) {
    require( depositor_ != address(0), "Invalid address" );

    // ...

    // depositor info is stored
    bondInfo[ depositor_ ] = Bond({
        payout: bondInfo[ depositor_ ].payout.add( payout ),
        vesting: terms.vestingTerm,
        lastTime: uint32(block.timestamp),
        pricePaid: priceInUSD
    });

    // ...
}
```

DESCRIPTION	The *depositor_* address is set by the UI and could have an incorrect value. This may cause users to transfer their funds to the wrong address. Note that projects have been hacked from the UI side in the past as well.
RECOMMENDATION	Limit the deposit function to specific contracts such as a zapper contract. These contracts should ensure that *msg.sender* is the same as the *depositor_*.
RESOLUTION	The project team has implemented the recommendation.

Users Redeeming Can Be Blocked Indefinitely

FINDING ID	#0003
SEVERITY	Low Risk
STATUS	Closed
LOCATION	BondDepositoryV2.sol -> 294-320

```
1  function redeem( address recipient_, bool stake_ ) external returns (
   uint ) {
2      Bond memory info = bondInfo[ recipient_ ];
3      // (seconds since last interaction / vesting term remaining)
4      uint percentVested = percentVestedFor( recipient_ );
5      uint payout;
6      uint remaining;
7      if ( percentVested >= 1e4 ) { // if fully vested
8          delete bondInfo[ recipient_ ]; // delete user info
9          payout = info.payout;
10         remaining = 0;
11     } else { // if unfinished
12         // calculate payout vested
13         payout = info.payout.mul( percentVested ).div( 1e4 );
14         // store updated deposit info
15         bondInfo[ recipient_ ] = Bond({
16             payout: info.payout.sub( payout ),
17             vesting: uint32( info.vesting.sub( uint32(
   block.timestamp ).sub( info.lastTime ) ) ),
18             lastTime: uint32(block.timestamp),
19             pricePaid: info.pricePaid
20         });
21
22         remaining = bondInfo[recipient_].payout;
23     }
24
25     emit BondRedeemed( recipient_, info.payout, 0 ); // emit bond
   data
26     return stakeOrSend( recipient_, stake_, info.payout ); // pay
   user everything due
27 }
```

DESCRIPTION	Redeem never checks that the caller is the same as *recipient_* thus a malicious actor could call redeem for another user. If they do this regularly, the user's tokens can be stuck in warmup.
RECOMMENDATION	Check that *recipient_* is the same as the caller.
RESOLUTION	The project team has implemented the recommendation.

High Upper Bound On Fees

FINDING ID	#0004
SEVERITY	Informational
STATUS	Closed
LOCATION	BondDepositoryV2.sol -> 154-173

```
1     function setBondTerms ( PARAMETER parameter_, uint input_ )
  external onlyOwner( )
2     {
3         // ...
4         else if ( parameter_ == PARAMETER.FEE ) { // 2
5             require( input_ <= 1e4, "DAO fee cannot exceed payout" );
6             terms.fee = input_;
7         }
8         // ...
9     }
```

DESCRIPTION	The maximum fee is 100%.
RECOMMENDATION	Reduce the maximum fee.
RESOLUTION	The project team lowered the max fee to 50%.

Static Analysis

No Events Emitted For Changes To Protocol Values

FINDING ID	#0005
SEVERITY	Informational
STATUS	Closed
LOCATION	• BondDepositoryV2.sol -> 154-173: *function setBondTerms (PARAMETER parameter_, uint input_) external onlyOwner();* • BondDepositoryV2.sol -> 182-198: *function setAdjustment (bool addition_,uint increment_,uint target_,uint32 buffer_) external onlyOwner();* • BondDepositoryV2.sol -> 204-207: *function setStaking(address staking_) external onlyOwner();* • BondDepositoryV2.sol -> 213-216: *function setStakingHelper(address stakingHelper_) external onlyOwner();*

DESCRIPTION	Functions that change important variables should emit events such that users can more easily monitor the change.
RECOMMENDATION	Emit events from these functions.
RESOLUTION	Events and emits for these variables are now added.

Missing Variable In Emit

FINDING ID	#0006
SEVERITY	Informational
STATUS	Closed
LOCATION	BondDepositoryV2.sol -> 28

```
1     event BondRedeemed( address indexed recipient, uint payout, uint
  remaining );
```

LOCATION	BondDepositoryV2.sol -> 294-320

```
1 function redeem( address recipient_, bool stake_ ) external returns (
  uint ) {
2     //...
3
4     emit BondRedeemed( recipient_, info.payout, 0 ); // emit bond
  data
5     return stakeOrSend( recipient_, stake_, info.payout ); // pay
  user everything due
6 }
```

DESCRIPTION	The variable *remaining* is not emitted in the event.
RECOMMENDATION	Add the *remaining* variable to the emit.
RESOLUTION	The project team implemented the recommended changes.

Contract Name Not Matching Filename

FINDING ID	#0007
SEVERITY	Informational
STATUS	Closed
LOCATION	• BondDepositoryV2.sol -> 15: *contract SecureBondDepositoryV2 is Ownable {*

DESCRIPTION	The file name and contract name don't match.
RECOMMENDATION	Change the file/contract names to match each other.
RESOLUTION	The filename has been changed to SecureBondDepositoryV2.sol

On-Chain Analysis

No Timelock For The Price Feed

FINDING ID	#0008
SEVERITY	Medium Risk
STATUS	Mitigated
LOCATION	ChainlinkBondingCalculator 0x30cA22ae3DCCB95DAf32693e8E1A63a7E77C4D69

DESCRIPTION	No timelock was provided for the owner address in ChainlinkBondingCalculator. The *updatePriceFeed()* function could therefore be called by a malicious actor as the contract owner to change the price feed to a malicious one EOA: 0x30cA22ae3DCCB95DAf32693e8E1A63a7E77C4D69
RECOMMENDATION	Change the owner to the timelock.
RESOLUTION	Ownership was transferred to the timelock. Note: timelock delay is 12 hours. Timelock 0xac4220abfd028f9c12b7916235180bbe73619b00

Low Minimum Timelock Delay

FINDING ID	#0009
SEVERITY	Low Risk
STATUS	Mitigated
LOCATION	SecureBondDepositoryV2 0x19456E31168a87C6Db420EF58aA0B57E7a3E8903

DESCRIPTION	The owner address is a 12-hour OpenZeppelin timelock controlled by a 3-5 multisig.
RECOMMENDATION	Obelisk recommends a timelock delay of at least 72 hours.
RESOLUTION	Project Team Comment: "FYI the timelock is only 12hrs, we understand that the recommendation is 72hrs and that 12hrs may be additional risk 👍"

External Addresses

External Contracts

These contracts are not part of the audit scope.

DAO

ADDRESS	0x82BAB147F3F8afbA380eDBE1792a7a71e2c9cb88
USAGE	0x19456E31168a87C6Db420EF58aA0B57E7a3E8903 *SecureBondDepositoryV2.DAO* - Immutable
IMPACT	• receives transfer of tokens deposited by users

SCR Token

ADDRESS	0x8183C18887aC4386CE09Dbdf5dF7c398DAcB2B5a
USAGE	0x19456E31168a87C6Db420EF58aA0B57E7a3E8903 *SecureBondDepositoryV2.SCR* - Immutable
IMPACT	• ERC20 Token

Timelock Contract

ADDRESS	0xac4220abfd028f9c12b7916235180bbe73619b00
USAGE	0x19456E31168a87C6Db420EF58aA0B57E7a3E8903 *SecureBondDepositoryV2.owner* - Variable *ChainlinkBondingCalculator.owner* - Variable
IMPACT	• has elevated permissions as owner, operator, or other

Principle Token

ADDRESS	0x21be370d5312f44cb42ce377bc9b8a0cef1a4c83
USAGE	0x19456E31168a87C6Db420EF58aA0B57E7a3E8903 *SecureBondDepositoryV2.principle* - Immutable
IMPACT	• ERC20 Token

Staking Contract

ADDRESS	0x3d97040e407078823891c59bb07eadb2ddf3ae32
USAGE	0x19456E31168a87C6Db420EF58aA0B57E7a3E8903 *SecureBondDepositoryV2.staking* - Variable
IMPACT	• impacts ability to deposit or withdraw tokens

Appendix A - Reviewed Documents

Document	Address
interfaces/bondingcalculator.sol	N/A
interfaces/erc20.sol	N/A
interfaces/staking.sol	N/A
interfaces/treasury.sol	N/A
FixedPoint.sol	N/A
ChainlinkBondingCalculator.sol	0x30cA22ae3DCCB95DAf32693e8E1A63a7E77C4D69
SecureBondDepositoryV2.sol Renamed in Rev 2 from: BondDepositoryV2.sol	0x19456E31168a87C6Db420EF58aA0B57E7a3E8903

Revisions

Revision 1	7f5bdda562e5d0528c8197cf8134fcbe11082c1f
Revision 2	9f8fdf86313acff977b69849947e865a75e81a6f
Revision 3	c535c99341d9eac8256b19ace94d6efbe37d6649

Imported Contracts

OpenZeppelin	3.4.2

Appendix B - Risk Ratings

Risk	Description
High Risk	A fatal vulnerability that can cause the loss of all Tokens / Funds.
Medium Risk	A vulnerability that can cause the loss of some Tokens / Funds.
Low Risk	A vulnerability that can cause the loss of protocol functionality.
Informational	Non-security issues such as functionality, style, and convention.

Appendix C - Finding Statuses

Closed	Contracts were modified to permanently resolve the finding.
Mitigated	The finding was resolved by other methods such as revoking contract ownership. The issue may require monitoring, for example in the case of a time lock.
Partially Closed	Contracts were updated to fix the issue in some parts of the code.
Partially Mitigated	Fixed by project-specific methods which cannot be verified on-chain. Examples include compounding at a given frequency.
Open	The finding was not addressed.

Appendix D - Audit Procedure

A typical Obelisk audit uses a combination of the three following methods:

Manual analysis consists of a direct inspection of the contracts to identify any security issues. Obelisk auditors use their experience in software development to spot vulnerabilities. Their familiarity with common contracts allows them to identify a wide range of issues in both forked contracts as well as original code.

Static analysis is software analysis of the contracts. Such analysis is called “static” as it examines the code outside of a runtime environment. Static analysis is a powerful tool used by auditors to identify subtle issues and to verify the results of manual analysis.

On-chain analysis is the audit of the contracts as they are deployed on the blockchain. This procedure verifies that:

- deployed contracts match those which were audited in manual/static analysis;
- contract values are set to reasonable values;
- contracts are connected so that interdependent contracts function correctly;
- and the ability to modify contract values is restricted via a timelock or DAO mechanism. (We recommend a timelock value of at least 72 hours)

Each obelisk audit is performed by at least two independent auditors who perform their analysis separately.

After the analysis is complete, the auditors will make recommendations for each issue based on best practices and industry standards. The project team can then resolve the issues, and the auditors will verify that the issues have been resolved with no new issues introduced.

Our auditing method lays a particular focus on the following important concepts:

- Quality code and the use of best practices, industry standards, and thoroughly tested libraries.
- Testing the contract from different angles to ensure that it works under a multitude of circumstances.
- Referencing the contracts through databases of common security flaws.

Follow Obelisk Auditing for the Latest Information

 ObeliskOrg

 ObeliskOrg

OBELISK

Part of Tibereum Group