SALUS SECURITY

MAY 2023

CODE SECURITY ASSESSMENT

TPROTOCOL

Overview

Project Summary

- Name: TProtocol
- Version: commit 51ead5d
- Platform: Ethereum
- Language: Solidity
- Repository: https://github.com/TProtocol/TBT
- Audit Scope: See Appendix - 1

Project Dashboard

Application Summary

Name	TProtocol
Version	v2
Type	Solidity
Dates	May 23 2023
Logs	May 17 2023; May 23 2023

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	5
Total informational issues	5
Total	10

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Centralization risk	Low	Centralization	Acknowledged
2	Allowing anyone to initialize on the implementation contract	Low	Business Logic	Acknowledged
3	Allowing to transfer under pause status	Low	Business Logic	Acknowledged
4	Missing events	Low	Logging	Acknowledged
5	Locked MPFee	Low	Business Logic	Acknowledged
6	Redundant code	Informational	Redundancy	Acknowledged
7	Can use immutable to save gas	Informational	Gas Optimization	Acknowledged
8	Can add a local variable instead of reading a storage multiple times	Informational	Gas Optimization	Acknowledged
9	Struct can be packed	Informational	Gas Optimization	Acknowledged
10	Spelling error	Informational	Code Quality	Acknowledged

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Centralization risk

Severity: Low	Category: Centralization
Target: - contracts/wTBTPoolV2Permission.sol	

Description

In wTBTPoolV2Permission contract, multiple roles such as ADMIN_ROLE, POOL_MANAGER_ROLE, APR_MANAGER_ROLE exist.

Even though, all roles are set to one address.

```
function initialize(
    ...
) public initializer {
    ...
    _setupRole(DEFAULT_ADMIN_ROLE, admin);
    _setRoleAdmin(ADMIN_ROLE, ADMIN_ROLE);
    _setRoleAdmin(POOL_MANAGER_ROLE, ADMIN_ROLE);
    _setRoleAdmin(APR_MANAGER_ROLE, ADMIN_ROLE);

    _setupRole(ADMIN_ROLE, admin);
    _setupRole(POOL_MANAGER_ROLE, admin);
    _setupRole(APR_MANAGER_ROLE, admin);
    ...
}
```

And most importantly, the DEFAULT_ADMIN_ROLE can:

- Add or remove an address to the previously mentioned roles

If an attacker were to gain access to the private keys associated with this role, they could cause significant damage to the project.

Recommendation

Consider setting each role to a different address.
It is recommended to transfer privileged roles to multi-signature accounts.

Status

This issue has been acknowledged by the team. Currently, the DEFAULT_ADMIN_ROLE and ADMIN_ROLE are only granted to address 0x6d2d493616e9e8407509e77c6f21f5f5f52199d1, which is a Timelock contract managed by a multi-sig wallet.

2. Allowing anyone to initialize on the implementation contract

Severity: Low	Category: Business Logic
Target: - contracts/TBT.sol - contracts/wTBTPoolV2Permission.sol	

Description

These contracts are upgradeable smart contracts and have an initialize() function. However, nothing prevents users from directly calling the initialize function on the contract implementation. According to the OpenZeppelin guidelines, the _disableInitializers function call should be added to the constructor to lock contracts after deployment.

```
function initialize() public initializer {

}
```

Recommendation

Consider adding a constructor with the _disableInitializers function call.

```
/// @custom:oz-upgrades-unsafe-allow constructor
constructor() {
    _disableInitializers();
}
```

Status

This issue has been acknowledged by the team.

3. Allowing to transfer under pause status

Severity: Low	Category: Business Logic
Target: - contracts/wTBTPoolV2Permission.sol	

Description

When the contract is paused, the token can still be transferred.

Recommendation

Consider adding the whenNotPaused modifier in the _transfer function if you wish to disallow fund-related transfers during the paused state.

Example:

```
function _transfer(address from, address to, uint256 amount) internal whenNotPaused
override {...}
```

Status

This issue has been acknowledged by the team.

4. Missing events

Severity: Low	Category: Logging
Target: - contracts/Treasury.sol - contracts/wTBTPoolV2Permission.sol	

Description

The contracts are missing events when key values are updated.

In the Treasury contract:
- setVault
- setMintPool
- setRedeemPool
- setCurvePool
- setMintThreshold
- setRedeemThreshold
- setPegPrice

In the wTBTPoolV2Permission contracts:
- setVault
- setTreasury
- setFeeCollector
- setManagementFeeCollector
- setTargetAPR
- setProcessPeriod
- setCapitalLowerBound
- setMintFeeRate
- setMintInterestCostFeeRate
- setRedeemFeeRate
- setRedeemMPFeeRate
- setManagementFeeRate

Recommendation

Consider adding variable update events.

Example:

```
event UpdateTargetAPR(uint256 oldTargetAPR, uint256 TargetAPR);
```

Status

This issue has been acknowledged by the team.

5. Locked MPFee

Severity: Low	Category: Business Logic

Target:
- contracts/wTBTPoolV2Permission.sol

Description

Even if MPFee is collected through the redeem() function, there is no part to withdraw the MPFee in the redeemUnderlyingTokenById() function.

This may cause MPFee to be locked in the wTBTPoolV2Permission contract.

contracts/wTBTPoolV2Permission.sol:L482-L510

```
function redeem(uint256 amount) external whenNotPaused realizeReward nonReentrant {
        ...
        uint256 redeemFeeAmount =
underlyingAmount.mul(redeemFeeRate).div(FEE_COEFFICIENT);
        uint256 redeemMPFeeAmount =
underlyingAmount.mul(redeemMPFeeRate).div(FEE_COEFFICIENT);
        uint256 amountAfterFee =
underlyingAmount.sub(redeemFeeAmount).sub(redeemMPFeeAmount);

        redeemIndex++;
        redeemDetails[redeemIndex] = RedeemDetail({
                id: redeemIndex,
                timestamp: block.timestamp,
                user: msg.sender,
                underlyingAmount: underlyingAmount,
                redeemAmountAfterFee: amountAfterFee,
                MPFee: redeemMPFeeAmount,
                protocolFee: redeemFeeAmount,
                isDone: false
        });
        ...
}
```

contracts/wTBTPoolV2Permission.sol:L558-L576

```
function redeemUnderlyingTokenById(uint256 _id) external whenNotPaused nonReentrant {
        ...
    vault.withdrawToUser(msg.sender, redeemAmountAfterFee);
    vault.withdrawToUser(feeCollector, protocolFee);
        ...
}
```

Recommendation

Consider adding a part to take MPFee in the redeemUnderlyingTokenById() function.

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

6. Redundant code

Severity: Informational	Category: Redundancy

Target:
- contracts/TBT.sol
- contracts/wTBTPoolV2Permission.sol

Description

These functions are all basic functions in ERC20, there is no need to rewrite them:

- contracts/TBT.sol:L120 - L123

```
function transfer(address _recipient, uint256 _amount) public override returns (bool)
{
    _transfer(msg.sender, _recipient, _amount);
    return true;
}
```

- contracts/wTBTPoolV2Permission:L584 - L585

```
function decimals() public pure override returns (uint8) {
       return 18;
}
```

- contracts/wTBTPoolV2Permission:L612 - L631

```
function transfer(address to, uint256 amount) public override returns (bool) {
    address owner = _msgSender();
    _transfer(owner, to, amount);
    return true;
}


function allowance(address owner, address spender) public view override returns
(uint256) {
    return super.allowance(owner, spender);
}


function approve(address spender, uint256 amount) public override returns (bool) {
    return super.approve(spender, amount);
}


function transferFrom(address from, address to, uint256 amount) public override
returns (bool) {
    address spender = _msgSender();
    _spendAllowance(from, spender, amount);
    _transfer(from, to, amount);
    return true;
}
```

The code is redundant in these places:

- contracts/wTBTPoolV2Permission:L17

```
import "hardhat/console.sol";
```

Redundant and ineffective checks:

- contracts/wTBTPoolV2Permission:L484,L532

```
require(totalUnderlying >= 0, "101");
```

There are still TODO comments in these places:

- contracts/wTBTPoolV2Permission:L77

```
// TODO: Can omit this, and calculate it from event.
uint256 public totalPendingRedeems;
```

- contracts/wTBTPoolV2Permission:L148

```
// TODO: revisit.
_setupRole(DEFAULT_ADMIN_ROLE, admin);
```

- contracts/wTBTPoolV2Permission:L378

```
// TODO: If use totalUnderlying, then the interest also incurs interest, do we want
to switch to principal?
return targetAPR.mul(totalUnderlying).div(365 days).div(APR_COEFFICIENT);
```

Recommendation

Consider removing the mentioned code and comments above.

Status

This issue has been acknowledged by the team.

7. Can use immutable to save gas

Severity: Informational	Category: Gas Optimization
Target: - contracts/Vault.sol - contracts/Treasury.sol	

Description

In the vault contract, underlying, stbt, and recovery could be made immutable.

- contracts/Vault:L15, L17, L19

```
IERC20 public underlying;
// STBT token address
IERC20 public stbt;
// recovery fund wallet
address public recovery;
```

In the Treasury contract, stbt, underlying, basis, recovery, and priceFeed could be made immutable.

- contracts/Treasury:L28, L30, L40, L44, L46

```
IERC20 public stbt;
// underlying token address
IERC20 public underlying;
```

```
uint256 public basis;
// recovery fund wallet
address public recovery;
// priceFeed be using check USDC is pegged
AggregatorInterface public priceFeed;
```

Recommendation

Consider setting the variables immutable.

Status

This issue has been acknowledged by the team.

8. Can add a local variable instead of reading a storage multiple times

Severity: Informational	Category: Gas Optimization

Target:
- contracts/wTBTPoolV2Permission.sol

Description

totalUnderlying is read multiple times in the redeem() and flashRedeem() functions.

```
function redeem(uint256 amount) external whenNotPaused realizeReward nonReentrant {
    require(amount <= cTokenBalances[msg.sender], "100");
    require(totalUnderlying >= 0, "101");
    require(cTokenTotalSupply > 0, "104");

    uint256 underlyingAmount = amount.mul(totalUnderlying).div(cTokenTotalSupply);

    require(totalUnderlying.sub(underlyingAmount) >= capitalLowerBound, "102");

    _burn(msg.sender, amount);
    totalUnderlying = totalUnderlying.sub(underlyingAmount);
    ...
}
```

```
function flashRedeem(
    uint256 amount,
    int128 j,
    uint256 minReturn
    ) external whenNotPaused realizeReward nonReentrant {
    require(amount <= cTokenBalances[msg.sender], "100");
    require(totalUnderlying >= 0, "101");
    require(cTokenTotalSupply > 0, "104");

    uint256 underlyingAmount = amount.mul(totalUnderlying).div(cTokenTotalSupply);

    require(totalUnderlying.sub(underlyingAmount) >= capitalLowerBound, "102");

    _burn(msg.sender, amount);
    totalUnderlying = totalUnderlying.sub(underlyingAmount);
    ...
}
```

Recommendation

Consider adding a local variable for totalUnderlying.

Status

This issue has been acknowledged by the team.

9. Struct can be packed

Severity: Informational	Category: Gas Optimization

Target:
- contracts/wTBTPoolV2Permission.sol

Description

The RedeemDetail struct could be packed by moving the user element either after or before isDone.

- contracts/wTBTPoolV2Permission.sol:L98 - L108

```
struct RedeemDetail {
    uint256 id;
    uint256 timestamp;
    address user;
    uint256 underlyingAmount;
    uint256 redeemAmountAfterFee;
    uint256 MPFee;
    uint256 protocolFee;
    // False not redeem, or True.
    bool isDone;
}
```

Recommendation

Consider moving the user element either after or before isDone.
Example:

```
struct RedeemDetail {
    uint256 id;
    uint256 timestamp;
    uint256 underlyingAmount;
    uint256 redeemAmountAfterFee;
    uint256 MPFee;
    uint256 protocolFee;
    // False not redeem, or True.
    bool isDone;
    address user;
}
```

Status

This issue has been acknowledged by the team.

10. Spelling error

Severity: Informational	Category: Code Quality
Target: - contracts/Vault.sol - contracts/Treasury.sol	

Description

The contracts have typo in some comments.

- contracts/Vault:L37, L39, L46, L48 (amout)

```
* @dev Transfer a give amout of underlying to user
* @param user user address
* @param amount the amout of underlying
```

```
* @dev Transfer a give amout of stbt to matrixport's mint pool
* @param mpRedeemPool user address
* @param stbtAmount the amout of underlying
```

- contracts/Treasury:L184, L185, L197, L198 (amout)

```
* @dev Transfer a give amout of stbt to matrixport's mint pool
* @param amount the amout of underlying token
```

Recommendation

Should be amount instead of amout.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 51ead5d:

File	SHA-1 hash
contracts\TBT.sol	00e12f8f915e79d6882e00f5dc9652effe25a0c7
contracts\Treasury.sol	5224280f6e271b5f6a53356050ac66098a255ea3
contracts\Vault.sol	1673ec9e3607307ee5b9cc87222ecc2aa2ad9704
contracts\wTBTPoolV2Permission.sol	f491db65eb4709014035e8fdacbf257243adabd3
contracts\tools\DomainAware.sol	addcf3bbc7b48c318d124f4faad5d38b7346fcb6
contracts\interface\ITBT.sol	a59699dd26ccfc3cd6233ca1187d8a72f855695e
contracts\interface\ITreasury.sol	03b312a20c8f1ffbb4c5a63bbade582538723eec
contracts\interface\IVault.sol	8f6dc88755b0ee1bc04f37ecf3566d273a136914
contracts\interface\IwTBTPoolV2Permission.sol	9d5db77eeb806e929dee95316fd48d893dc375cd
contracts\interface\ICurve.sol	a0e5187803bfdc3927efab3a32995924426c209c
contracts\interface\AggregatorInterface.sol	eebc603d5ab7a21155521e4374bc5b200b686e00