Competitive Security Assessment

Shield Staking Vault P2

Mar 15th, 2023

Secure3

secure3.io

Summary

This report is prepared for the project to identify vulnerabilities and issues in the smart contract source code. A group of NDA covered experienced security experts have participated in the Secure3's Audit Contest to find vulnerabilities and optimizations. Secure3 team has participated in the contest process as well to provide extra auditing coverage and scrutiny of the finding submissions.

The comprehensive examination and auditing scope includes:

- Cross checking contract implementation against functionalities described in the documents and white paper disclosed by the project owner.
- Contract Privilege Role Review to provide more clarity on smart contract roles and privilege.
- Using static analysis tools to analyze smart contracts against common known vulnerabilities patterns.
- Verify the code base is compliant with the most up-to-date industry standards and security best practices.
- Comprehensive line-by-line manual code review of the entire codebase by industry experts.

The security assessment resulted in findings that are categorized in four severity levels: Critical, Medium, Low, Informational. For each of the findings, the report has included recommendations of fix or mitigation for security and best practices.

Overview

Project Detail

Project Name	Shield Staking Vault P2
Platform & Language	Solidity
Codebase	• https://github.com/ShieldDAODev/shield-staking-vault-v1 • audit commit - 4338af5d9e2494a7689e023aa60cd3086c299fd6 • final commit - a562ab5158eb8e3b01deb716788d3c61f4698e17
Audit Methodology	• Audit Contest • Business Logic and Code Review • Privileged Roles Review • Static Analysis

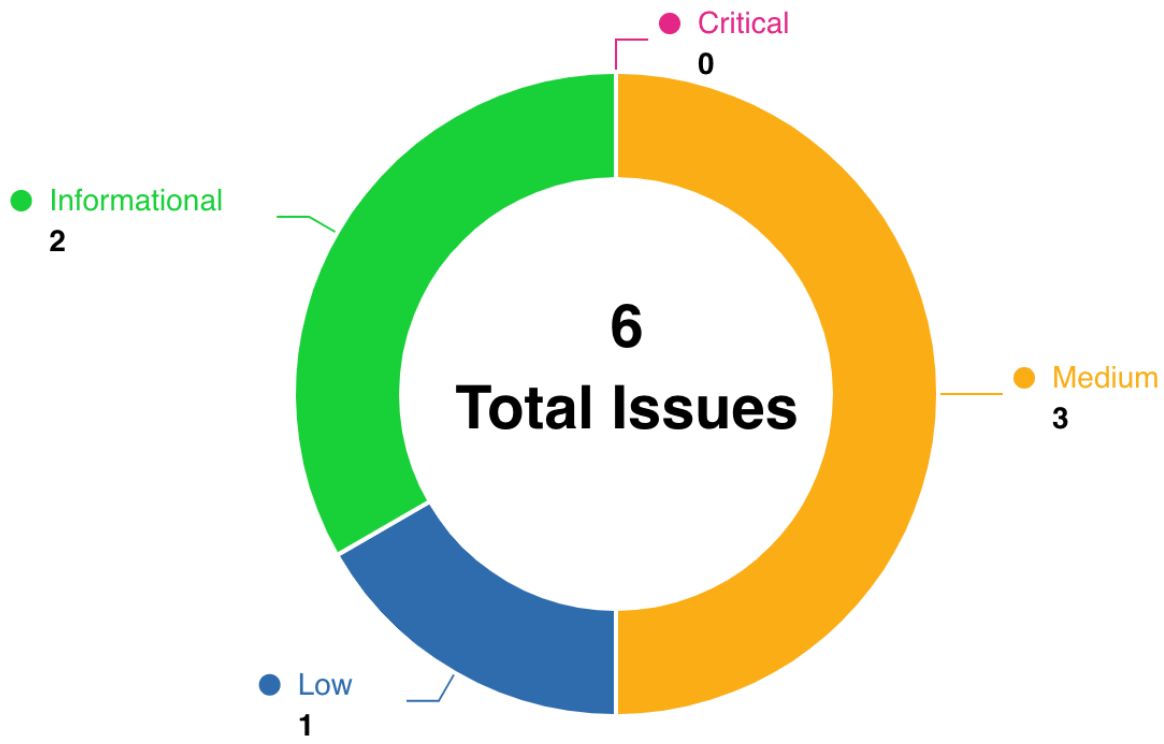
Code Vulnerability Review Summary

Vulnerability Level	Total	Reported	Acknowledged	Fixed	Mitigated	Declined
Critical	0	0	0	0	0	0
Medium	3	0	0	1	0	2
Low	1	0	0	1	0	0
Informational	2	0	1	1	0	0

Audit Scope

File	Commit Hash
contracts/StakingVault.sol	4338af5d9e2494a7689e023aa60cd3086c299fd6
contracts/VaultManager.sol	4338af5d9e2494a7689e023aa60cd3086c299fd6
contracts/interfaces/IStableSwap.sol	4338af5d9e2494a7689e023aa60cd3086c299fd6
contracts/structs/VaultStorage.sol	4338af5d9e2494a7689e023aa60cd3086c299fd6

Code Assessment Findings



ID	Name	Category	Severity	Status	Contributor
SSV-1	Events not emitted for important state changes	Logical	Informational	Acknowledged	Secure3
SSV-2	Logic error in `StakingVault` contract `getAllEtherInvested` function	Logical	Medium	Declined	w2ning
SSV-3	Missing error message in require statements	Code Style	Informational	Fixed	Secure3
SSV-4	Redundant logic in `StakingVault` contract `settlement` function	Logical	Low	Fixed	alansh

SSV-5	**Variable `broker` can never be changed in `StakingVault` contract**	**Logical**	**Medium**	**Fixed**	**w2ning**
SSV-6	**`settlement` function is vulnerable to MEV attack**	**Logical**	**Medium**	**Declined**	**comcat**

SSV-1:Events not emitted for important state changes

Category	Severity	Code Reference	Status	Contributor
Logical	Informational	• code/contracts/StakingVault.sol#L804	Acknowledged	Secure3

Code

```
804:    function setCurveEnable(bool _flag, uint256 _minReceived)
```

Description

Secure3 : When critical state variables are changed, events are not emitted. This is against the best practice.

Recommendation

Secure3 : Emit an event to track the event. Consider below sample

```
event setCurveEnable(bool _flag, uint256 _minReceived, uint256 time);

function setCurveEnable(bool _flag, uint256 _minReceived)
        ...
        emit setCurveEnable(_flag, _minReceived, block.timestamp);
}
```

Client Response

Acknowledged.

SSV-2:Logic error in `StakingVault` contract `getAllEtherInvested` function

Category	Severity	Code Reference	Status	Contributor
Logical	Medium	• code/contracts/StakingVault.sol#L716	Declined	w2ning

Code

```
716:        uint256 ldoBalance = ICurveGauge(CURVE_GAUGE).claimable_reward(
```

Description

w2ning : Only the amount of claimable LDO Token is read, not real claim. Cannot use so many tokens to swap on exchange.

```
//Only the amount of claimable LDO Tokens is read here, not real cliam
uint256 ldoBalance = ICurveGauge(CURVE_GAUGE).claimable_reward(
    address(this),
    LDO_TOKEN
);

ldoBalance = ldoBalance.add(ERC20(LDO_TOKEN).balanceOf(address(this)));

//Cannot swap so many LDO Tokens
if (ldoBalance > 0) {
    etherOut = etherOut.add(
        IQuoter(QUOTER).quoteExactInputSingle(
            LDO_TOKEN,
            WETH,
            3000,
            ldoBalance,
            0
        )
    );
}
```

Recommendation

w2ning : Call `claim_Rewards` function before swap

Consider below fix in the `StakingVault.getAllEtherInvested()` function

```
uint256 claimable = ICurveGauge(CURVE_GAUGE).claimable_reward(
    address(this),
    LDO_TOKEN
);

if (claimable > 0) {

  // Fix: Call `claim_Rewards` function before swap
  ICurveGauge(CURVE_GAUGE).claim_rewards();
}

// Fix: The amount to swap shall be the balance of LDO Token on this contract
ldoBalance = ERC20(LDO_TOKEN).balanceOf(address(this));

//Cannot swap so much LDO Tokens
if (ldoBalance > 0) {
    etherOut = etherOut.add(
        IQuoter(QUOTER).quoteExactInputSingle(
            LDO_TOKEN,
            WETH,
            3000,
            ldoBalance,
            0
        )
    );
}
```

Client Response

Declined. In the contract of `Curve Gauge`, both `claim_rewards()` and `claimable_reward()` funtion executes the logic of claiming reward tokens through `_checkpoint_rewards`. From a logical perspective, the amount returned by `claimable_reward` is the same as the actual amount of LDO claimed by `claim_rewards`.
Ref:https://etherscan.io/address/0x182b723a58739a9c974cfdb385ceadb237453c28#code

SSV-3:Missing error message in require statements

Category	Severity	Code Reference	Status	Contributor
Code Style	Informational	• code/contracts/StakingVault.sol#L809	Fixed	Secure3

Code

```
809:        require(_flag != curveEnabled);
```

Description

Secure3 : require can be used to check for conditions and throw an exception if the condition is not met, in which case the error message provided by the developer will appear. This is why a very descriptive error message is needed.

```
    require(_flag != curveEnabled);
```

Recommendation

Secure3 : Adding an error message describing the failed condition

Client Response

Fixed

SSV-4:Redundant logic in `StakingVault` contract `settlement` function

Category	Severity	Code Reference	Status	Contributor
Logical	Low	• code/contracts/StakingVault.sol#L516-L518	Fixed	alansh

Code

```
516:                } else {
517:                    steNeeded = steNeeded > steBalance ? steNeeded : steBalance;
518:                }
```

Description

alansh :

```
                uint256 steNeeded;
                uint256 steBalance = ERC20(STETH).balanceOf(address(this));

                if (!isSellAll) {
                   ....
                } else {
                    steNeeded = steNeeded > steBalance ? steNeeded : steBalance;
                }
```

since there is no other assignment for `steNeeded` before the else block and `steNeeded` will be 0 and `steNeeded > steBalance` always be False, hence it can be simplified to `steNeeded = steBalance` in the else block. Please double check the logic if this is desired

Recommendation

alansh : Apply the above change. Or change it this way to avoid query `balanceOf` when `!isSellAll`:

```
uint256 steNeeded;

if (!isSellAll) {
    steNeeded = VaultMath.getMinSellAmount(
        assets.etherNeedToSell,
        MULTIPLIER,
        lp_slippage
    );
} else {
    steNeeded = ERC20(STETH).balanceOf(address(this));
}
```

Client Response

Fixed

SSV-5:Variable `broker` can never be changed in `StakingVault` contract

Category	Severity	Code Reference	Status	Contributor
Logical	Medium	• code/contracts/StakingVault.sol#L148	Fixed	w2ning

Code

```
148:        if (broker != address(0) && _broker != address(0)) {
```

Description

w2ning : The variable `broker` can never be changed.

Because the variable `broker` is always empty, the function `addBrokerRelationship` can never be called.

```
function deposit(address _broker)
    external
    payable
    nonReentrant
    notTerminated
    notExpired
{
    _depositFor(msg.value, msg.sender);

    // The variable 'broker' is always empty
    if (broker != address(0) && _broker != address(0)) {
        // The function 'addBrokerRelationship' can never be called
        IBroker(broker).addBrokerRelationship(_broker, msg.sender);
    }
}
```

Recommendation

w2ning : Consider below fix in the `StakingVault.initialize()` function

```
function initialize(
      ...
      _broker,
      ...
      ){

      ...
      broker = _broker;
      ...

}
```

Client Response

Fixed by removing the unused `addBrokerRelationship()` call.

SSV-6: `settlement` function is vulnerable to MEV attack

Category	Severity	Code Reference	Status	Contributor
Logical	Medium	• code/contracts/StakingVault.sol#L482	Declined	comcat

Code

```
482:                    lpNeeded = IStableSwap(STABLE_SWAP).calc_token_amount(
```

Description

comcat : Though the settlement function has the modifier `onlyGovernance`, which means it can only be called by governor, the way it implement trading is still vulnerable to MEV attack. For the `curveEnabled = true, isSellAll = true`, it just withdraws all the lp token from `CURVE_GAUGE`, and then call the `remove_liquidity_one_coin`. when calculate the min amount received, it use the `calc_withdraw_one_coin` function, which will calculate it on-chain. This means that if the pool STETH_ETH is imbalanced due to a large swap, the `calc_withdraw_one_coin` will gives the imbalanced answer, not the amount the protocol expected (close to 1:1). Hence the `_min_amount` (Minimum amount of the coin to receive) parameter passed in (`actualWithdraw`) will be useless for the protection of slippage.

```
if (assets.etherNeedToSell > 0) {
            if (curveEnabled) {
                ...
                if (isSellAll) {
                    lpNeeded = lpBalance;
                } else {
                ...
                }

                ICurveGauge(CURVE_GAUGE).withdraw(lpNeeded);
                uint256 actualWithdraw = IStableSwap(STABLE_SWAP)
                    .calc_withdraw_one_coin(lpNeeded, 0);
                IStableSwap(STABLE_SWAP).remove_liquidity_one_coin(
                    lpNeeded,
                    0,
                    actualWithdraw
                );
            }
```

Recommendation

comcat : add another params in `settlement` function, which will limit the min amount received by `remove_liquidity_one_coin` when sell lp.

Client Response

Declined. MEV attack is already prevented by below `require` statement, where the ETH amount obtained after settlement is compared with the external information `_minEtherReceived` passed into the function together with the balance snapshot `etherBefore` right before the stable swap.

```
        require(
            _minEtherReceived.add(etherBefore) <= address(this).balance,
            "N"
        );
```

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 Invoices, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Invoice. This report provided in connection with the services set forth in the Invoices shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Invoice. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Secure3’s prior written consent in each instance.

This report is not an “endorsement” or “disapproval” of any particular project or team. This report is not an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Secure3 to perform a security assessment. This report does not provide any warranty or guarantee of free of bug of codes analyzed, nor do they provide any indication of the technologies, business model or legal compliancy.

This report should not be used in any way to make decisions around investment or involvement with any particular project. Instead, it represents an extensive assessing process intending to help our customers increase the quality of their code and high-level consistency of implementation and business model, while reducing the risk presented by cryptographic tokens and blockchain technology.

Secure3’s position on the final decisions over blockchain technologies and corresponding associated transactions is that each company and individual are responsible for their own due diligence and continuous security.

The assessment services provided by Secure3 is subject to dependencies and under continuing development. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.