SALUS SECURITY

FEB 2024

CODE SECURITY ASSESSMENT

NATIVE

Overview

Project Summary

- Name: Native - Aqua
- Platform: EVM-compatible Chains
- Language: Solidity
- Repository:https://github.com/Native-org/native-contracts
- Audit Scope: See Appendix - 1

Project Dashboard

Application Summary

Name	Native - Aqua
Version	v2
Type	Solidity
Date	Feb 09 2024
Logs	Feb 07 2024; Feb 09 2024

Vulnerability Summary

Total High-Severity issues	1
Total Medium-Severity issues	4
Total Low-Severity issues	1
Total informational issues	3
Total	9

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	The trader may lose the collateral when removing the collateral	High	Business logic	Resolved
2	Use of deprecated Chainlink API	Medium	Business logic	Resolved
3	Possible reentrancy in AquaVault	Medium	Reentrancy	Resolved
4	Unchecked ERC20 transfers	Medium	Data Validation	Resolved
5	Centralization risk	Medium	Centralization	Acknowledged
6	Missing event	Low	Logging	Resolved
7	Use beacon instead of transparent proxy mode	Informational	Business logic	Acknowledged
8	Missing two-step transfer ownership pattern	Informational	Business logic	Acknowledged
9	Missing zero address checks	Informational	Data Validation	Acknowledged

2.2 Notable Findings

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

1. The trader may lose the collateral when removing the collateral	
Severity: High	Category: Business logic
Target: - contracts/Aqua/AquaVault.sol	

Description

contracts/Aqua/AquaVault.sol

```
function removeCollateral(
    RemoveCollateralRequest calldata request,
    bytes calldata signature,
    address recipient
) external onlyTraderOrSettler(request.trader) {
    aquaVaultSignatureCheck.verifyRemoveCollateralSignature(request, signature, signer);

    TokenAmountUint[] calldata tokens = request.tokens;
    AquaVaultLogic.removeCollateral(tokens, recipient, request.trader, aquaCollateral);

    emit CollateralRemoved(request.trader, tokens);
}
```

In the removeCollateral() function, the recipient parameter is not validated. This means that as long as the request parameters are valid, the caller can arbitrarily choose the recipient to receive the collateral.

An attacker only needs to detect a removeCollateral transaction in the transaction pool that has not yet been confirmed in a block. By changing the recipient parameter to their own address and providing a higher gas fee, the attacker can front-run the transaction.

This allows an attacker to take collateral from a legitimate trader.

And a similar attack scenario exists with the settle() function.

Recommendation

It is recommended that the recipient be part of the signature rather than specified by the caller. Alternatively qualify the caller as request.trader or settler.

Status

The team has resolved this issue in commit 876c4c9, 2019e4c.

2. Use of deprecated Chainlink API	
Severity: Medium	Category:Business logic
Target: - contracts/Aqua/ChainlinkPriceOracle.sol	

Description

According to Chainlink's documentation, the latestAnswer function is deprecated. This function does not error if no answer has been reached but returns 0, causing an incorrect price feed to the Aqua.

Recommendation

It is recommended to use the latestRoundData function to get the price instead. Add checks on the return data with proper revert messages if the price is stale or the round is uncomplete, for example:

```
function getUnderlyingPrice(CToken cToken) external view override returns (uint) {
    ..
    (uint80 roundID, int256 price,, uint256 updatedAt, uint80 answeredInRound) =
feed.latestRoundData();
    require(updatedAt != 0);
    require(answeredInRound >= roundID);
    require(updatedAt >= block.timestamp - maxDelayTime);
    require(price_ > 0, "LibChainlinkPrice: price cannot be negative");
    ..
}
```

Status

The team has resolved this issue in commit 57a4c2c.

3. Possible reentrancy in AquaVault	
Severity: Medium	Category: Reentrancy
Target: - contracts/libraries/AquaVaultLogic.sol	

Description

contracts/libraries/AquaVaultLogic.sol:L120-L151

```
function settle(
    IAquaVault.TokenAmountInt[] memory positionUpdates,
    address trader,
    address recipient,
    mapping(address => mapping(address => int)) storage positions,
    mapping(address => AquaLpToken) storage lpTokens
) external {
    ...
    if (amount > 0) {
        IERC20(token).safeTransferFrom(msg.sender, address(this), uint(amount));
    } else {
        IERC20(token).safeTransfer(recipient, uint(-amount));
    }

    lpTokens[token].updateNetBorrow(-int(amount));
    positions[trader][token] = newPositionValue;
}
```

Since the tokens are transferred before the position and borrowing values are updated, it is possible to perform a reentrancy attack if the token has some kind of call-back functionality, e.g. ERC777. pBTC is an ERC777 token that is currently available on Native.

And a similar attack scenario exists with the removeCollateral() function.

Recommendation

It is recommended to follow the "Check-Effect-Interaction" rule in the code.

Status

The team has resolved this issue in commit 36fa4b4, 2bac51f.

4. Unchecked ERC20 transfers	
Severity: Medium	Category: Data Validation
Target: - contracts/Aqua/AquaLpToken.sol - contracts/Aqua/AquaVault.sol	

Description

In the code base, there are Multiple calls to transferFrom and transfer are frequently done without checking the results. For certain ERC20 tokens, if insufficient tokens are present, no revert occurs but a result of "false" is returned.

This may result in a situation where the transfer fails but the execution is successful. In this case, either the user or the project may suffer a loss of funds.

Compatibility with these non-standard ERC-20 is considered a best practice (as AquaLpToken.doTransferIn() does).

Recommendation

Consider using the SafeERC20 library implementation from OpenZeppelin and call safeTransfer or safeTransferFrom when transferring ERC20 tokens. Also, we recommend aligning the processing of transfers in the transferOut function with that of the transferIn function.

Status

The team has resolved this issue in commit 5173271, 6d5a826

5. Centralization risk	
Severity: Medium	Category: Centralization
Target: - contracts/Aqua/AquaVault.sol	

Description

In the AquaVault contract, there is a privileged admin role. The admin role has the ability to:

- set nativePool address
- set signature check address
- set tokens allowance
- set trader's settler
- set trader
- set liquidator
- set signer
- set epoch updater
- new market

If the admin's private key is compromised, the attacker can exploit the admin's role to change the signer's address and maliciously modify and liquidate the positions of regular users in the roles of settler or liquidator. In this scenario, users' assets would be severely compromised.

Recommendation

We recommend transferring the admin role to a multisig account with a timelock feature for enhanced security. This ensures that no single person has full control over the account and that any changes must be authorized by multiple parties.

Status

This issue has been acknowledged by the team.

6. Missing event	
Severity: Low	Category: Logging
Target: - contracts/Aqua/AquaVault.sol - contracts/Aqua/AquaLpToken.sol - contracts/NativeRfqPool.sol - contracts/NativePool.sol	

Description

Events allow capturing the changed parameters so that off-chain tools/interfaces can register such changes that allow users to evaluate them. Missing events do not promote transparency and if such changes immediately affect users' perception of fairness or trustworthiness, they could exit the protocol causing a reduction in protocol users.
In Native, the following functions are missing events:

- AquaVault.setAdmin()
- AquaVault.setNativePool()
- AquaVault.setSignatureCheck()
- AquaVault.setTraderSettler()
- AquaVault.setTrader()
- AquaVault.setLiquidator()
- AquaVault.setSigner()
- AquaVault.setEpochUpdater()
- AquaVault.supportMarket()
- AquaLpToken.setAquaVault()
- AquaLpToken.updateNetBorrow()
- AquaLpToken.updateReserve()
- NativeRfqPool.setTreasury()
- NativeRfqPool.updateSigner()
- NativeRfqPool.setPostTradeCallback()
- NativeRfqPool.setPause()
- NativePoolFactory.setPoolImplementation()
- NativePoolFactory.setRegistry()
- NativePool.setPauser()

Recommendation

It is recommended to emit events for critical state changes.

Status

The team has resolved this issue in commit b54fc4a.

2.3 Informational Findings

7. Use beacon instead of transparent proxy mode	
Severity: Informational	Category: Business logic
Target: - contracts/NativePoolFactory.sol	

Description

contracts/NativePoolFactory.sol:L90-L119

```
function createNewPool(
    NewPoolConfig calldata poolConfig
) external override whenNotPaused nonReentrant returns (address) {
    ...
    address pool = address(
        new ERC1967Proxy(poolImplementation, abi.encodeWithSelector(INIT_SELECTOR,
poolConfig, registry))
    );
    ...
}
```

The createNewPool() function uses a transparent proxy to create a new pool implementation.
contracts/NativePoolFactory.sol:L137-L139

```
function setPoolImplementation(address newPoolImplementation) external override
onlyOwner {
    poolImplementation = newPoolImplementation;
}
```

When the owner updates the implementation of the pool by calling the setPoolImplementation function, the deployed proxy contracts' implementations will not be automatically updated. This implies that you need to upgrade each contract individually, which is not conducive to project management, as there may be differences between the implementations of the old and new pools.

Recommendation

It is recommended to use beacon proxy mode to create the pool.

Status

This issue has been acknowledged by the team.

8. Missing two-step transfer ownership pattern	
Severity: Informational	Category: Business logic
Target: - contracts/NativePoolFactory.sol	

Description

The NativePoolFactory contract inherits from the Ownable contract. This contract does not implement a two-step process for transferring ownership. Thus, ownership of the contract can easily be lost when making a mistake in transferring ownership.

Recommendation

Consider using the Ownable2Step contract from OpenZeppelin instead.

Status

This issue has been acknowledged by the team.

9. Missing zero address checks	
Severity: Informational	Category: Data Validation
Target: - contracts/Aqua/AquaVault.sol - contracts/NativePoolFactory.sol - contracts/NativeRfqPool.sol	

Description

It is considered a security best practice to verify addresses against the zero address during initialisation or setting.

However, this precautionary step is missing in the following function:

- AquaVault.setAdmin()
- AquaVault.setNativePool()
- AquaVault.setSigner()
- AquaVault.setEpochUpdater()
- NativePoolFactory.setPauser()
- NativePoolFactory.setPoolImplementation()
- NativeRfqPool.updateSigner()
- NativeRfqPool.setTreasury()

Recommendation

Consider adding zero address checks for these functions.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 01feeb6:

File	SHA-1 hash
NativeRouter.sol	b293712dd46e565919a7dc517199a96f956cfb1a
NativePoolFactory.sol	2423c4f992b983b379a58b9fb1c99bee91b7c1cd
NativeRfqPool.sol	8f67755c1cb48f0ba9e3aa59010f759ef22716b9
AquaLpToken.sol	4c9cc4f18e8ff60dce6966a68da71db76d596ed3
AquaVault.sol	d311bc2eb3c353755e288b022af891677eb0bda1
AquaVaultSignatureCheck.sol	11b3a94123462bc4c8601a790fecc5d3b50e84b5
ChainlinkPriceOracle.sol	51bbbb5a52fbc18cee6fa7b6408de6bd76f1e329