SHELLBOXES

NAOS Finance

Smart Contract Security Audit

Prepared by ShellBoxes

February 28th, 2022 - April 15th, 2022

Shellboxes.com

contact@shellboxes.com

Document Properties

Client	NAOS Finance
Version	Public
Classification	Public

Scope

The NAOS Finance Contract in the NAOS Finance Repository

Repo	Commit Hash
https://github.com/NAOS-Finance/olympus-contracts	1b38de119010541958519cdc652515a847aa3927

Files	MD5 Hash
OlympusAuthority.sol	82f5cc94c95aa576ec7a37c963891e79
custom/CustomBond.sol	780b3cef0cc6cab2ef8b58a528d144ed
custom/CustomTreasury.sol	e18a992fb8b5d0f45c8147aff2f14fef

Re-Audit

Repo	Commit Hash
https://github.com/NAOS-Finance/olympus-contracts	c514923408afa61183bd30296e4cbd719718e7e8

Files	MD5 Hash
OlympusAuthority.sol	26e4091f59bd7253a7719beacdd5ea04
custom/CustomBond.sol	3b39a63c1aa2665fdd00813c177e024e
custom/CustomTreasury.sol	e18a992fb8b5d0f45c8147aff2f14fef

Contacts

COMPANY	EMAIL
ShellBoxes	contact@shellboxes.com

Contents

1 Introduction

NAOS Finance engaged ShellBoxes to conduct a security assessment on the NAOS Finance beginning on February 28th, 2022 and ending April 15th, 2022. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About NAOS Finance

NAOS Finance is a decentralized real-world asset (RWA) lending protocol that facilitates the borrowing of crypto native assets by using RWA as collateral. NAOS has established a large network of corporate borrowers and is operating with financing licenses in multiple regions. In the effort of connecting CeFi with DeFi, NAOS takes an ecosystem approach and looks to engage in meaningful strategic partnerships to expand the boundary of decentralized finance.

Issuer	NAOS Finance
Website	`ttps://naos.finance`
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

ShellBoxes used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

1.2.1 Risk Methodology

Vulnerabilities or bugs identified by ShellBoxes are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.

Impact	High	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Low
		High	Medium	Low

Likelihood

2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the NAOS Finance implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include , 2 high-severity, 2 medium-severity, 3 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
Missing Transfer Verification	HIGH	Fixed
The Policy Have Super Control Over The Treasury	HIGH	Acknowledged
Restriction Can Be Bypassed	MEDIUM	Acknowledged
Possible Desynchronization In The Deposit Function	MEDIUM	Acknowledged
Missing Address Verification	LOW	Fixed
Missing Address Verification	LOW	Fixed
Floating Pragma	LOW	Fixed

3 Finding Details

A CustomBond.sol

A.1 Missing Transfer Verification [HIGH]

Description:

The ERC20 standard token implementation functions return the transaction status as a boolean. It is a good practice to check for the return status of the function call to ensure that the transaction was successful. It is the developer's responsibility to enclose these function calls with require() to ensure that, when the intended ERC20 function call returns false, the caller transaction also fails. However, it is mostly missed by developers when they carry out checks in effect, the transaction would always succeed, even if the token transfer did not.

Code:

Listing 1: CustomBond.sol

```
if ( percentVested >= 10000 ) { // if fully vested
    delete bondInfo[ _depositor ]; // delete user info
    emit BondRedeemed( _depositor, info.payout, 0 ); // emit bond data
    payoutToken.transfer( _depositor, info.payout );
    return info.payout;

} else { // if unfinished
    // calculate payout vested
    uint payout = info.payout.mul( percentVested ).div( 10000 );

    // store updated deposit info
    bondInfo[ _depositor ] = Bond({
        payout: info.payout.sub( payout ),
        vesting: info.vesting.sub( block.number.sub( info.lastBlock ) ),
        lastBlock: block.number,
        truePricePaid: info.truePricePaid
    });

    emit BondRedeemed( _depositor, payout, bondInfo[ _depositor ].payout
        ↪ );
    payoutToken.transfer( _depositor, payout );
    return payout;
}
```

Risk Level:

Likelihood – 2
Impact - 5

Recommendation:

Use the safeTransfer function from the safeERC20 Implementation, or put the transfer call inside an assert or require to verify that the transfer has passed successfully.

Status - Fixed

The NAOS team has fixed the issue by wrapping the transfer call inside a require statement to make sure the transfer has passed successfully.

A.2 Restriction Can Be Bypassed [MEDIUM]

Description:

The setAdjustment() function contains a restriction which ensures that the policy cannot increment the rate with a value more than 3 of the previous one.The restriction can be bypassed if the policy calls the setAdjustment() with the addition argument true then calls the deposit() function in order to execute the adjust() multiple times. In that way, in every call of the adjust function, the terms.controlVariable will be incremented by the rate value. Therefore, the policy can increment the controlVariable multiple times and bypass the restriction.

Code:

Listing 2: CustomBond.sol

```solidity
function setAdjustment (
    bool _addition,
    uint _increment,
    uint _target,
    uint _buffer
) external onlyPolicy {
    require( _increment <= terms.controlVariable.mul( 30 ).div( 1000 ),
        ↪ "Increment too large" );

    adjustment = Adjust({
        add: _addition,
        rate: _increment,
        target: _target,
        buffer: _buffer,
        lastBlock: block.number
    });
}
```

Listing 3: CustomBond.sol

```solidity
function adjust() internal {
    uint blockCanAdjust = adjustment.lastBlock.add( adjustment.buffer );
    if( adjustment.rate != 0 && block.number >= blockCanAdjust ) {
        uint initial = terms.controlVariable;
        if ( adjustment.add ) {
            terms.controlVariable = terms.controlVariable.add( adjustment
                ↪ .rate );
            if ( terms.controlVariable >= adjustment.target ) {
                adjustment.rate = 0;
            }
        } else {
            terms.controlVariable = terms.controlVariable.sub( adjustment
                ↪ .rate );
            if ( terms.controlVariable <= adjustment.target ) {
                adjustment.rate = 0;
            }
        }
        adjustment.lastBlock = block.number;
        emit ControlVariableAdjustment( initial, terms.controlVariable,
            ↪ adjustment.rate, adjustment.add );
    }
}
```

Risk Level:

Likelihood – 3
Impact - 2

Recommendation:

It's recommended to remove the restriction as it can be bypassed or document this behavior.

Status - Acknowledged

The NAOS team has acknowledged the risk, mentioning that the blockCanAdjust parameter in adjust() function can prevent parameter over-tuning.

A.3 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, some of the contract's functionality may become inaccessible.

Code:

Listing 4: CustomBond.sol

```
constructor(
    address _customTreasury,
    address _payoutToken,
    address _principalToken,
    address _initialOwner
) OlympusAccessControlled(IOlympusAuthority(_initialOwner)) {
    require( _customTreasury != address(0) );
    customTreasury = ICustomTreasury( _customTreasury );
    require( _payoutToken != address(0) );
    payoutToken = IERC20Metadata( _payoutToken );
    require( _principalToken != address(0) );
    principalToken = IERC20Metadata( _principalToken );
}
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It is recommended to verify that the addresses provided in the arguments are different from the address(0).

Status - Fixed

The NAOS team has fixed the issue by adding a require statement to the OlympusAccess-Controlled's constructor to make sure the argument is different from the address(0).

B CustomTreasury.sol

B.1 The Policy Have Super Control Over The Treasury [HIGH]

Description:

Using the withdraw() function, the policy can send any amount of any token to whatever destination from the treasury balance. This represents a significant centralization risk where the policy have too much control over the treasury.

Code:

Listing 5: CustomTreasury.sol

```
function withdraw(address _token, address _destination, uint _amount)
    ↪ external onlyPolicy {
    IERC20Metadata(_token).safeTransfer(_destination, _amount);

    emit Withdraw(_token, _destination, _amount);
}
```

Risk Level:

Likelihood – 3
Impact - 5

Recommendation:

It is recommended to use a multisig wallet in order to avoid centralization risks.

Status - Acknowledged

The NAOS team has acknowledged the risk, mentioning that a multisig wallet will be used as policy to prevent centralization risk.

B.2 Possible Desynchronization In The Deposit Function [MEDIUM]

Description:

The policy can add a new CustomBond contract, this new contract is able to call the CustomTreasury directly. Thus, the CustomBond can call the deposit function, this behavior will cause a desynchronization between the bond and treasury contracts.

Code:

Listing 6: CustomBond.sol

```
principalToken.approve( address(customTreasury), _amount );
customTreasury.deposit( address(principalToken), _amount, payout );
```

Listing 7: CustomTreasury.sol

```
function toggleBondContract(address _bondContract) external onlyPolicy {
    bondContract[_bondContract] = !bondContract[_bondContract];
    emit BondContractToggled(_bondContract, bondContract[_bondContract])
        ↪ ;
}
```

Listing 8: CustomTreasury.sol

```
function deposit(address _principleTokenAddress, uint
    ↪ _amountPrincipleToken, uint _amountPayoutToken) external {
    require(bondContract[msg.sender], "msg.sender is not a bond contract
        ↪ ");
    IERC20Metadata(_principleTokenAddress).safeTransferFrom(msg.sender,
        ↪ address(this), _amountPrincipleToken);
    IERC20Metadata(payoutToken).safeTransfer(msg.sender,
        ↪ _amountPayoutToken);
}
```

Risk Level:

Likelihood – 2
Impact - 4

Recommendation:

It is recommended to store the bytecode hash of the CustomBond contract in the CustomTreasury, then in the toggleBondContract() function extract the bytecode hash of the bondContract and verify if it is the same as the one stored in the treasury contract.

Status - Acknowledged

The NAOS team has acknowledged the risk, saying the policy multisig will remediate the risk.

Listing 9: Extract Byte Code Hash

```
function extractByteCode(address _addr) internal view returns (bytes
    ↪ memory o_code) {
    assembly {
        // retrieve the size of the code, this needs assembly
        let size := extcodesize(_addr)
        // allocate output byte array - this could also be done without
            ↪ assembly
        // by using o_code = new bytes(size)
        o_code := mload(0x40)
        // new "memory end" including padding
        mstore(0x40, add(o_code, and(add(add(size, 0x20), 0x1f), not(0x1f
            ↪ ))))
        // store length in memory
        mstore(o_code, size)
        // actually retrieve the code, this needs assembly
        extcodecopy(_addr, add(o_code, 0x20), 0, size)
    }
}
```

```
function extractByteCodeHash(address _addr) internal view returns (
    ↪ bytes32){
    return keccak256(extractByteCode(_addr));
}
```

Listing 10: CustomTreasury.sol

```
function toggleBondContract(address _bondContract) external onlyPolicy {
    require(BondContractHash == extractByteCodeHash(_bondContract) , "
        ↪ This is not a Bond contract");
    bondContract[_bondContract] = !bondContract[_bondContract];
    mit BondContractToggled(_bondContract, bondContract[_bondContract]);
}
```

C OlympusAuthority.sol

C.1 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, some of the contract's functionality may become inaccessible.

Code:

Listing 11: OlympusAuthority.sol

```
constructor(
    address _governor,
    address _guardian,
    address _policy,
    address _vault
) OlympusAccessControlled(IOlympusAuthority(address(this))) {
    governor = _governor;
    emit GovernorPushed(address(0), governor, true);
    guardian = _guardian;
    emit GuardianPushed(address(0), guardian, true);
    policy = _policy;
    emit PolicyPushed(address(0), policy, true);
    vault = _vault;
    emit VaultPushed(address(0), vault, true);
}
```

Listing 12: OlympusAuthority.sol

```
function pushGovernor(address _newGovernor, bool _effectiveImmediately)
    ↪ external onlyGovernor {
    if (_effectiveImmediately) governor = _newGovernor;
    newGovernor = _newGovernor;
    emit GovernorPushed(governor, newGovernor, _effectiveImmediately);
}
```

Listing 13: OlympusAuthority.sol

```
function pushGuardian(address _newGuardian, bool _effectiveImmediately)
    ↪ external onlyGovernor {
    if (_effectiveImmediately) guardian = _newGuardian;
    newGuardian = _newGuardian;
    emit GuardianPushed(guardian, newGuardian, _effectiveImmediately);
}
```

Listing 14: OlympusAuthority.sol

```
    function pushPolicy(address _newPolicy, bool _effectiveImmediately)
        ↪ external onlyGovernor {
        if (_effectiveImmediately) policy = _newPolicy;
        newPolicy = _newPolicy;
        emit PolicyPushed(policy, newPolicy, _effectiveImmediately);
    }
```

Listing 15: OlympusAuthority.sol

```
    function pushVault(address _newVault, bool _effectiveImmediately)
        ↪ external onlyGovernor {
```

```
        if (_effectiveImmediately) vault = _newVault;
        newVault = _newVault;
        emit VaultPushed(vault, newVault, _effectiveImmediately);
    }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It is recommended to verify that the addresses provided in the arguments are different from the address(0).

Status - Fixed

The NAOS team has fixed the issue by adding a require statement to make sure the arguments are different from the address(0).

C.2 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.7.5. Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps to ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 16: OlympusAuthority.sol

```
// SPDX-License-Identifier: AGPL-3.0
pragma solidity >=0.7.5;
```

Risk Level:

Likelihood – 2
Impact - 1

Recommendation:

Consider locking the pragma version. It is advised that floating pragma not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Fixed

The NAOS team has fixed the issue by locking the pragma version to 0.7.5.

4 Best Practices

BP.1 Use The Latest Solidity Version

Description:

The contract makes use of the pragma version 0.7.5. There are newer versions that include some breaking changes such as : overflow protection, revert opcode after failing assertions and other internal checks like division by zero or arithmetic overflow, explicit conversions between literals and address having the type address instead of address payable, ... Therefore, It is recommended to use the latest versions of solidity to make use of the new functionalities.

Code:

Listing 17: CustomBond.sol

```
// SPDX-License-Identifier: AGPL-3.0-or-later
pragma solidity 0.7.5;
```

Listing 18: CustomTreasury.sol

```
// SPDX-License-Identifier: AGPL-3.0-or-later
pragma solidity 0.7.5;
```

5 Static Analysis (Slither)

Description:

ShellBoxes expanded the coverage of the specific contract areas using automated testing methodologies. Slither, a Solidity static analysis framework, was one of the tools used. Slither was run on all-scoped contracts in both text and binary formats. This tool can be used to test mathematical relationships between Solidity instances statically and variables that allow for the detection of errors or inconsistent usage of the contracts' APIs throughout the entire codebase.

Results:

```
//CustomBond.sol

CustomBond.redeem(address) (contracts/custom/CustomBond.sol#228-255)
    ↪ ignores return value by payoutToken.transfer(_depositor,info.
    ↪ payout) (contracts/custom/CustomBond.sol#235)
CustomBond.redeem(address) (contracts/custom/CustomBond.sol#228-255)
    ↪ ignores return value by payoutToken.transfer(_depositor,payout) (
    ↪ contracts/custom/CustomBond.sol#251)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unchecked-transfer

FullMath.fullDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.
    ↪ sol#12-31) performs a multiplication on the result of a division:
         -d /= pow2 (contracts/libraries/FullMath.sol#18)
         -r *= 2 - d * r (contracts/libraries/FullMath.sol#22)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.
    ↪ sol#12-31) performs a multiplication on the result of a division:
         -d /= pow2 (contracts/libraries/FullMath.sol#18)
         -r *= 2 - d * r (contracts/libraries/FullMath.sol#23)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.
    ↪ sol#12-31) performs a multiplication on the result of a division:
        -d /= pow2 (contracts/libraries/FullMath.sol#18)
        -r *= 2 - d * r (contracts/libraries/FullMath.sol#24)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.
    ↪ sol#12-31) performs a multiplication on the result of a division:
        -d /= pow2 (contracts/libraries/FullMath.sol#18)
        -r *= 2 - d * r (contracts/libraries/FullMath.sol#25)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.
    ↪ sol#12-31) performs a multiplication on the result of a division:
        -d /= pow2 (contracts/libraries/FullMath.sol#18)
        -r *= 2 - d * r (contracts/libraries/FullMath.sol#26)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.
    ↪ sol#12-31) performs a multiplication on the result of a division:
        -d /= pow2 (contracts/libraries/FullMath.sol#18)
        -r *= 2 - d * r (contracts/libraries/FullMath.sol#27)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.
    ↪ sol#12-31) performs a multiplication on the result of a division:
        -d /= pow2 (contracts/libraries/FullMath.sol#18)
        -r *= 2 - d * r (contracts/libraries/FullMath.sol#28)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.
    ↪ sol#12-31) performs a multiplication on the result of a division:
        -d /= pow2 (contracts/libraries/FullMath.sol#18)
        -r *= 2 - d * r (contracts/libraries/FullMath.sol#29)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.
    ↪ sol#12-31) performs a multiplication on the result of a division:
        -l /= pow2 (contracts/libraries/FullMath.sol#19)
        -l * r (contracts/libraries/FullMath.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #divide-before-multiply

Reentrancy in CustomBond.deposit(uint256,uint256,address) (contracts/
    ↪ custom/CustomBond.sol#177-222):
        External calls:
        - principalToken.safeTransferFrom(msg.sender,address(this),
            ↪ _amount) (contracts/custom/CustomBond.sol#198)
        - principalToken.approve(address(customTreasury),_amount) (
           ↪ contracts/custom/CustomBond.sol#199)
        - customTreasury.deposit(address(principalToken),_amount,payout)
           ↪ (contracts/custom/CustomBond.sol#200)
        State variables written after the call(s):
        - BondPriceChanged(_bondPrice(),debtRatio()) (contracts/custom/
           ↪ CustomBond.sol#215)
               - terms.minimumPrice = 0 (contracts/custom/CustomBond.sol
                  ↪ #299)
        - adjust() (contracts/custom/CustomBond.sol#220)
               - terms.controlVariable = terms.controlVariable.add(
                  ↪ adjustment.rate) (contracts/custom/CustomBond.sol
                  ↪ #267)
               - terms.controlVariable = terms.controlVariable.sub(
                  ↪ adjustment.rate) (contracts/custom/CustomBond.sol
                  ↪ #272)
        - totalDebt = totalDebt.add(value) (contracts/custom/CustomBond.
           ↪ sol#203)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #reentrancy-vulnerabilities-1

CustomBond.deposit(uint256,uint256,address) (contracts/custom/CustomBond
   ↪ .sol#177-222) ignores return value by principalToken.approve(
   ↪ address(customTreasury),_amount) (contracts/custom/CustomBond.sol
   ↪ #199)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #unused-return

CustomBond.initializeBond(uint256,uint256,uint256,uint256,uint256,
   ↪ uint256) (contracts/custom/CustomBond.sol#100-121) should emit an
   ↪  event for:
        - totalDebt = _initialDebt (contracts/custom/CustomBond.sol#119)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #missing-events-arithmetic
```

```
Reentrancy in CustomBond.deposit(uint256,uint256,address) (contracts/
    ↪ custom/CustomBond.sol#177-222):
        External calls:
        - principalToken.safeTransferFrom(msg.sender,address(this),
            ↪ _amount) (contracts/custom/CustomBond.sol#198)
        - principalToken.approve(address(customTreasury),_amount) (
            ↪ contracts/custom/CustomBond.sol#199)
        - customTreasury.deposit(address(principalToken),_amount,payout)
            ↪ (contracts/custom/CustomBond.sol#200)
        State variables written after the call(s):
        - adjust() (contracts/custom/CustomBond.sol#220)
                - adjustment.rate = 0 (contracts/custom/CustomBond.sol
                    ↪ #269)
                - adjustment.rate = 0 (contracts/custom/CustomBond.sol
                    ↪ #274)
                - adjustment.lastBlock = block.number (contracts/custom/
                    ↪ CustomBond.sol#277)
        - bondInfo[_depositor] = Bond(bondInfo[_depositor].payout.add(
            ↪ payout),terms.vestingTerm,block.number,trueBondPrice()) (
            ↪ contracts/custom/CustomBond.sol#206-211)
        - totalPayoutGiven = totalPayoutGiven.add(payout) (contracts/
            ↪ custom/CustomBond.sol#218)
        - totalPrincipalBonded = totalPrincipalBonded.add(_amount) (
            ↪ contracts/custom/CustomBond.sol#217)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-2

Reentrancy in CustomBond.deposit(uint256,uint256,address) (contracts/
    ↪ custom/CustomBond.sol#177-222):
        External calls:
        - principalToken.safeTransferFrom(msg.sender,address(this),
            ↪ _amount) (contracts/custom/CustomBond.sol#198)
        - principalToken.approve(address(customTreasury),_amount) (
          ↪ contracts/custom/CustomBond.sol#199)
        - customTreasury.deposit(address(principalToken),_amount,payout)
          ↪ (contracts/custom/CustomBond.sol#200)
        Event emitted after the call(s):
        - BondCreated(_amount,payout,block.number.add(terms.vestingTerm))
          ↪  (contracts/custom/CustomBond.sol#214)
        - BondPriceChanged(_bondPrice(),debtRatio()) (contracts/custom/
          ↪ CustomBond.sol#215)
        - ControlVariableAdjustment(initial,terms.controlVariable,
          ↪ adjustment.rate,adjustment.add) (contracts/custom/
          ↪ CustomBond.sol#278)
                - adjust() (contracts/custom/CustomBond.sol#220)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

Address.isContract(address) (contracts/libraries/Address.sol#23-34) uses
    ↪  assembly
        - INLINE ASM (contracts/libraries/Address.sol#30-32)
Address._functionCallWithValue(address,bytes,uint256,string) (contracts/
    ↪ libraries/Address.sol#139-165) uses assembly
        - INLINE ASM (contracts/libraries/Address.sol#157-160)
Address._verifyCallResult(bool,bytes,string) (contracts/libraries/
    ↪ Address.sol#223-244) uses assembly
        - INLINE ASM (contracts/libraries/Address.sol#236-239)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

CustomBond.initializeBond(uint256,uint256,uint256,uint256,uint256,
    ↪ uint256) (contracts/custom/CustomBond.sol#100-121) compares to a
    ↪ boolean constant:
        -require(bool,string)(initialized == false,initialized) (
          ↪ contracts/custom/CustomBond.sol#108)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #boolean-equality

Different versions of Solidity are used:
        - Version used: ['0.7.5', '>=0.7.5', '^0.7.5']
        - 0.7.5 (contracts/custom/CustomBond.sol#2)
        - 0.7.5 (contracts/interfaces/ICustomTreasury.sol#2)
        - >=0.7.5 (contracts/interfaces/IERC20.sol#2)
        - >=0.7.5 (contracts/interfaces/IERC20Metadata.sol#2)
        - >=0.7.5 (contracts/interfaces/IOlympusAuthority.sol#2)
        - >=0.7.5 (contracts/libraries/Address.sol#2)
        - ^0.7.5 (contracts/libraries/FixedPoint.sol#2)
        - ^0.7.5 (contracts/libraries/FullMath.sol#2)
        - >=0.7.5 (contracts/libraries/SafeERC20.sol#2)
        - ^0.7.5 (contracts/libraries/SafeMath.sol#2)
        - >=0.7.5 (contracts/types/OlympusAccessControlled.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #different-pragma-directives-are-used

Address._functionCallWithValue(address,bytes,uint256,string) (contracts/
    ↪ libraries/Address.sol#139-165) is never used and should be
    ↪ removed
Address._verifyCallResult(bool,bytes,string) (contracts/libraries/
    ↪ Address.sol#223-244) is never used and should be removed
Address.functionCall(address,bytes) (contracts/libraries/Address.sol
    ↪ #78-80) is never used and should be removed
Address.functionCall(address,bytes,string) (contracts/libraries/Address.
    ↪ sol#88-94) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/
    ↪ libraries/Address.sol#107-113) is never used and should be
    ↪ removed
Address.functionCallWithValue(address,bytes,uint256,string) (contracts/
    ↪ libraries/Address.sol#125-137) is never used and should be
    ↪ removed
```

```
Address.functionDelegateCall(address,bytes) (contracts/libraries/Address
    ↪ .sol#201-203) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (contracts/libraries/
    ↪ Address.sol#211-221) is never used and should be removed
Address.functionStaticCall(address,bytes) (contracts/libraries/Address.
    ↪ sol#173-175) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (contracts/libraries/
    ↪ Address.sol#183-193) is never used and should be removed
Address.isContract(address) (contracts/libraries/Address.sol#23-34) is
    ↪ never used and should be removed
Address.sendValue(address,uint256) (contracts/libraries/Address.sol
    ↪ #52-58) is never used and should be removed
Babylonian.sqrt(uint256) (contracts/libraries/FixedPoint.sol#7-48) is
    ↪ never used and should be removed
BitMath.mostSignificantBit(uint256) (contracts/libraries/FixedPoint.sol
    ↪ #52-84) is never used and should be removed
FixedPoint.decode(FixedPoint.uq112x112) (contracts/libraries/FixedPoint.
    ↪ sol#101-103) is never used and should be removed
FixedPoint.sqrt(FixedPoint.uq112x112) (contracts/libraries/FixedPoint.
    ↪ sol#126-134) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (contracts/libraries/
    ↪ SafeERC20.sol#35-45) is never used and should be removed
SafeERC20.safeTransfer(IERC20,address,uint256) (contracts/libraries/
    ↪ SafeERC20.sol#23-33) is never used and should be removed
SafeERC20.safeTransferETH(address,uint256) (contracts/libraries/
    ↪ SafeERC20.sol#47-51) is never used and should be removed
SafeMath.sqrrt(uint256) (contracts/libraries/SafeMath.sol#56-67) is
    ↪ never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Low level call in Address.sendValue(address,uint256) (contracts/
    ↪ libraries/Address.sol#52-58):
        - (success) = recipient.call{value: amount}() (contracts/
           ↪ libraries/Address.sol#56)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
    ↪ string) (contracts/libraries/Address.sol#125-137):
        - (success,returndata) = target.call{value: value}(data) (
           ↪ contracts/libraries/Address.sol#135)
Low level call in Address._functionCallWithValue(address,bytes,uint256,
    ↪ string) (contracts/libraries/Address.sol#139-165):
        - (success,returndata) = target.call{value: weiValue}(data) (
           ↪ contracts/libraries/Address.sol#148)
Low level call in Address.functionStaticCall(address,bytes,string) (
    ↪ contracts/libraries/Address.sol#183-193):
        - (success,returndata) = target.staticcall(data) (contracts/
           ↪ libraries/Address.sol#191)
Low level call in Address.functionDelegateCall(address,bytes,string) (
    ↪ contracts/libraries/Address.sol#211-221):
        - (success,returndata) = target.delegatecall(data) (contracts/
           ↪ libraries/Address.sol#219)
Low level call in SafeERC20.safeTransferFrom(IERC20,address,address,
    ↪ uint256) (contracts/libraries/SafeERC20.sol#10-21):
        - (success,data) = address(token).call(abi.encodeWithSelector(
           ↪ IERC20.transferFrom.selector,from,to,amount)) (contracts/
           ↪ libraries/SafeERC20.sol#16-18)
Low level call in SafeERC20.safeTransfer(IERC20,address,uint256) (
    ↪ contracts/libraries/SafeERC20.sol#23-33):
        - (success,data) = address(token).call(abi.encodeWithSelector(
           ↪ IERC20.transfer.selector,to,amount)) (contracts/libraries/
           ↪ SafeERC20.sol#28-30)
Low level call in SafeERC20.safeApprove(IERC20,address,uint256) (
    ↪ contracts/libraries/SafeERC20.sol#35-45):
        - (success,data) = address(token).call(abi.encodeWithSelector(
           ↪ IERC20.approve.selector,to,amount)) (contracts/libraries/
           ↪ SafeERC20.sol#40-42)
```

```
Low level call in SafeERC20.safeTransferETH(address,uint256) (contracts/
    ↪ libraries/SafeERC20.sol#47-51):
        - (success) = to.call{value: amount}(new bytes(0)) (contracts/
            ↪ libraries/SafeERC20.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Parameter CustomBond.initializeBond(uint256,uint256,uint256,uint256,
    ↪ uint256,uint256)._controlVariable (contracts/custom/CustomBond.
    ↪ sol#101) is not in mixedCase
Parameter CustomBond.initializeBond(uint256,uint256,uint256,uint256,
    ↪ uint256,uint256)._vestingTerm (contracts/custom/CustomBond.sol
    ↪ #102) is not in mixedCase
Parameter CustomBond.initializeBond(uint256,uint256,uint256,uint256,
    ↪ uint256,uint256)._minimumPrice (contracts/custom/CustomBond.sol
    ↪ #103) is not in mixedCase
Parameter CustomBond.initializeBond(uint256,uint256,uint256,uint256,
    ↪ uint256,uint256)._maxPayout (contracts/custom/CustomBond.sol#104)
    ↪  is not in mixedCase
Parameter CustomBond.initializeBond(uint256,uint256,uint256,uint256,
    ↪ uint256,uint256)._maxDebt (contracts/custom/CustomBond.sol#105)
    ↪ is not in mixedCase
Parameter CustomBond.initializeBond(uint256,uint256,uint256,uint256,
    ↪ uint256,uint256)._initialDebt (contracts/custom/CustomBond.sol
    ↪ #106) is not in mixedCase
Parameter CustomBond.setBondTerms(CustomBond.PARAMETER,uint256).
    ↪ _parameter (contracts/custom/CustomBond.sol#132) is not in
    ↪ mixedCase
Parameter CustomBond.setBondTerms(CustomBond.PARAMETER,uint256)._input (
    ↪ contracts/custom/CustomBond.sol#132) is not in mixedCase
Parameter CustomBond.setAdjustment(bool,uint256,uint256,uint256).
    ↪ _addition (contracts/custom/CustomBond.sol#152) is not in
    ↪ mixedCase
```

```
Parameter CustomBond.setAdjustment(bool,uint256,uint256,uint256).
    ↪ _increment (contracts/custom/CustomBond.sol#153) is not in
    ↪ mixedCase
Parameter CustomBond.setAdjustment(bool,uint256,uint256,uint256)._target
    ↪  (contracts/custom/CustomBond.sol#154) is not in mixedCase
Parameter CustomBond.setAdjustment(bool,uint256,uint256,uint256)._buffer
    ↪  (contracts/custom/CustomBond.sol#155) is not in mixedCase
Parameter CustomBond.deposit(uint256,uint256,address)._amount (contracts
    ↪ /custom/CustomBond.sol#177) is not in mixedCase
Parameter CustomBond.deposit(uint256,uint256,address)._maxPrice (
    ↪ contracts/custom/CustomBond.sol#177) is not in mixedCase
Parameter CustomBond.deposit(uint256,uint256,address)._depositor (
    ↪ contracts/custom/CustomBond.sol#177) is not in mixedCase
Parameter CustomBond.redeem(address)._depositor (contracts/custom/
    ↪ CustomBond.sol#228) is not in mixedCase
Parameter CustomBond.payoutFor(uint256)._value (contracts/custom/
    ↪ CustomBond.sol#347) is not in mixedCase
Parameter CustomBond.percentVestedFor(address)._depositor (contracts/
    ↪ custom/CustomBond.sol#390) is not in mixedCase
Parameter CustomBond.pendingPayoutFor(address)._depositor (contracts/
    ↪ custom/CustomBond.sol#407) is not in mixedCase
Struct FixedPoint.uq112x112 (contracts/libraries/FixedPoint.sol#88-90)
    ↪ is not in CapWords
Struct FixedPoint.uq144x112 (contracts/libraries/FixedPoint.sol#92-94)
    ↪ is not in CapWords
Parameter OlympusAccessControlled.setAuthority(IOlympusAuthority).
    ↪ _newAuthority (contracts/types/OlympusAccessControlled.sol#48) is
    ↪  not in mixedCase
Variable OlympusAccessControlled.UNAUTHORIZED (contracts/types/
    ↪ OlympusAccessControlled.sol#11) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions
```

```
CustomBond.maxPayout() (contracts/custom/CustomBond.sol#329-331) uses
   ↪ literals with too many digits:
       - payoutToken.totalSupply().mul(terms.maxPayout).div(100000) (
          ↪ contracts/custom/CustomBond.sol#330)
Babylonian.sqrt(uint256) (contracts/libraries/FixedPoint.sol#7-48) uses
   ↪ literals with too many digits:
       - xx >= 0x100000000000000000000000000000000 (contracts/libraries/
          ↪ FixedPoint.sol#12)
Babylonian.sqrt(uint256) (contracts/libraries/FixedPoint.sol#7-48) uses
   ↪ literals with too many digits:
       - xx >= 0x10000000000000000 (contracts/libraries/FixedPoint.sol
          ↪ #16)
Babylonian.sqrt(uint256) (contracts/libraries/FixedPoint.sol#7-48) uses
   ↪ literals with too many digits:
       - xx >= 0x100000000 (contracts/libraries/FixedPoint.sol#20)
BitMath.mostSignificantBit(uint256) (contracts/libraries/FixedPoint.sol
   ↪ #52-84) uses literals with too many digits:
       - x >= 0x100000000000000000000000000000000 (contracts/libraries/
          ↪ FixedPoint.sol#55)
BitMath.mostSignificantBit(uint256) (contracts/libraries/FixedPoint.sol
   ↪ #52-84) uses literals with too many digits:
       - x >= 0x10000000000000000 (contracts/libraries/FixedPoint.sol
          ↪ #59)
BitMath.mostSignificantBit(uint256) (contracts/libraries/FixedPoint.sol
   ↪ #52-84) uses literals with too many digits:
       - x >= 0x100000000 (contracts/libraries/FixedPoint.sol#63)
FixedPoint.slitherConstructorConstantVariables() (contracts/libraries/
   ↪ FixedPoint.sol#87-135) uses literals with too many digits:
       - Q112 = 0x10000000000000000000000000000 (contracts/libraries/
          ↪ FixedPoint.sol#97)
FixedPoint.slitherConstructorConstantVariables() (contracts/libraries/
   ↪ FixedPoint.sol#87-135) uses literals with too many digits:
       - Q224 = 0
          ↪ x100000000000000000000000000000000000000000000000000000000
```

```
        ↪  (contracts/libraries/FixedPoint.sol#98)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #too-many-digits

FixedPoint.Q224 (contracts/libraries/FixedPoint.sol#98) is never used in
    ↪  FixedPoint (contracts/libraries/FixedPoint.sol#87-135)
FixedPoint.LOWER_MASK (contracts/libraries/FixedPoint.sol#99) is never
    ↪ used in FixedPoint (contracts/libraries/FixedPoint.sol#87-135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-state-variable

OlympusAccessControlled.UNAUTHORIZED (contracts/types/
    ↪ OlympusAccessControlled.sol#11) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #state-variables-that-could-be-declared-constant
contracts/custom/CustomBond.sol analyzed (13 contracts with 78 detectors
    ↪ ), 85 result(s) found



//OlympusAuthority.sol

OlympusAuthority.constructor(address,address,address,address)._governor
    ↪ (contracts/OlympusAuthority.sol#30) lacks a zero-check on :
                - governor = _governor (contracts/OlympusAuthority.sol#35)
OlympusAuthority.constructor(address,address,address,address)._guardian
    ↪ (contracts/OlympusAuthority.sol#31) lacks a zero-check on :
                - guardian = _guardian (contracts/OlympusAuthority.sol#37)
OlympusAuthority.constructor(address,address,address,address)._policy (
    ↪ contracts/OlympusAuthority.sol#32) lacks a zero-check on :
                - policy = _policy (contracts/OlympusAuthority.sol#39)
OlympusAuthority.constructor(address,address,address,address)._vault (
    ↪ contracts/OlympusAuthority.sol#33) lacks a zero-check on :
                - vault = _vault (contracts/OlympusAuthority.sol#41)
```

```
OlympusAuthority.pushGovernor(address,bool)._newGovernor (contracts/
    ↪ OlympusAuthority.sol#47) lacks a zero-check on :
              - governor = _newGovernor (contracts/OlympusAuthority.sol
                ↪ #48)
              - newGovernor = _newGovernor (contracts/OlympusAuthority.
                ↪ sol#49)
OlympusAuthority.pushGuardian(address,bool)._newGuardian (contracts/
    ↪ OlympusAuthority.sol#53) lacks a zero-check on :
              - guardian = _newGuardian (contracts/OlympusAuthority.sol
                ↪ #54)
              - newGuardian = _newGuardian (contracts/OlympusAuthority.
                ↪ sol#55)
OlympusAuthority.pushPolicy(address,bool)._newPolicy (contracts/
    ↪ OlympusAuthority.sol#59) lacks a zero-check on :
              - policy = _newPolicy (contracts/OlympusAuthority.sol#60)
              - newPolicy = _newPolicy (contracts/OlympusAuthority.sol
                ↪ #61)
OlympusAuthority.pushVault(address,bool)._newVault (contracts/
    ↪ OlympusAuthority.sol#65) lacks a zero-check on :
              - vault = _newVault (contracts/OlympusAuthority.sol#66)
              - newVault = _newVault (contracts/OlympusAuthority.sol#67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Parameter OlympusAuthority.pushGovernor(address,bool)._newGovernor (
    ↪ contracts/OlympusAuthority.sol#47) is not in mixedCase
Parameter OlympusAuthority.pushGovernor(address,bool).
    ↪ _effectiveImmediately (contracts/OlympusAuthority.sol#47) is not
    ↪ in mixedCase
```

```
Parameter OlympusAuthority.pushGuardian(address,bool)._newGuardian (
    ↪ contracts/OlympusAuthority.sol#53) is not in mixedCase
Parameter OlympusAuthority.pushGuardian(address,bool).
    ↪ _effectiveImmediately (contracts/OlympusAuthority.sol#53) is not
    ↪ in mixedCase
Parameter OlympusAuthority.pushPolicy(address,bool)._newPolicy (
    ↪ contracts/OlympusAuthority.sol#59) is not in mixedCase
Parameter OlympusAuthority.pushPolicy(address,bool).
    ↪ _effectiveImmediately (contracts/OlympusAuthority.sol#59) is not
    ↪ in mixedCase
Parameter OlympusAuthority.pushVault(address,bool)._newVault (contracts/
    ↪ OlympusAuthority.sol#65) is not in mixedCase
Parameter OlympusAuthority.pushVault(address,bool)._effectiveImmediately
    ↪  (contracts/OlympusAuthority.sol#65) is not in mixedCase
Parameter OlympusAccessControlled.setAuthority(IOlympusAuthority).
    ↪ _newAuthority (contracts/types/OlympusAccessControlled.sol#48) is
    ↪  not in mixedCase
Variable OlympusAccessControlled.UNAUTHORIZED (contracts/types/
    ↪ OlympusAccessControlled.sol#11) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

OlympusAccessControlled.UNAUTHORIZED (contracts/types/
    ↪ OlympusAccessControlled.sol#11) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #state-variables-that-could-be-declared-constant
contracts/OlympusAuthority.sol analyzed (3 contracts with 78 detectors),
    ↪  20 result(s) found


//CustomTreasury.sol
```

```
Reentrancy in CustomTreasury.withdraw(address,address,uint256) (
    ↪ contracts/custom/CustomTreasury.sol#76-80):
        External calls:
        - IERC20Metadata(_token).safeTransfer(_destination,_amount) (
            ↪ contracts/custom/CustomTreasury.sol#77)
        Event emitted after the call(s):
        - Withdraw(_token,_destination,_amount) (contracts/custom/
            ↪ CustomTreasury.sol#79)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

Address.isContract(address) (contracts/libraries/Address.sol#23-34) uses
    ↪  assembly
        - INLINE ASM (contracts/libraries/Address.sol#30-32)
Address._functionCallWithValue(address,bytes,uint256,string) (contracts/
    ↪ libraries/Address.sol#139-165) uses assembly
        - INLINE ASM (contracts/libraries/Address.sol#157-160)
Address._verifyCallResult(bool,bytes,string) (contracts/libraries/
    ↪ Address.sol#223-244) uses assembly
        - INLINE ASM (contracts/libraries/Address.sol#236-239)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

Different versions of Solidity are used:
        - Version used: ['0.7.5', '>=0.7.5', '^0.7.5']
        - 0.7.5 (contracts/custom/CustomTreasury.sol#2)
        - >=0.7.5 (contracts/interfaces/IERC20.sol#2)
        - >=0.7.5 (contracts/interfaces/IERC20Metadata.sol#2)
        - >=0.7.5 (contracts/interfaces/IOlympusAuthority.sol#2)
        - >=0.7.5 (contracts/libraries/Address.sol#2)
        - >=0.7.5 (contracts/libraries/SafeERC20.sol#2)
        - ^0.7.5 (contracts/libraries/SafeMath.sol#2)
        - >=0.7.5 (contracts/types/OlympusAccessControlled.sol#2)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #different-pragma-directives-are-used

Address._functionCallWithValue(address,bytes,uint256,string) (contracts/
    ↪ libraries/Address.sol#139-165) is never used and should be
    ↪ removed
Address._verifyCallResult(bool,bytes,string) (contracts/libraries/
    ↪ Address.sol#223-244) is never used and should be removed
Address.functionCall(address,bytes) (contracts/libraries/Address.sol
    ↪ #78-80) is never used and should be removed
Address.functionCall(address,bytes,string) (contracts/libraries/Address.
    ↪ sol#88-94) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/
    ↪ libraries/Address.sol#107-113) is never used and should be
    ↪ removed
Address.functionCallWithValue(address,bytes,uint256,string) (contracts/
    ↪ libraries/Address.sol#125-137) is never used and should be
    ↪ removed
Address.functionDelegateCall(address,bytes) (contracts/libraries/Address
    ↪ .sol#201-203) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (contracts/libraries/
    ↪ Address.sol#211-221) is never used and should be removed
Address.functionStaticCall(address,bytes) (contracts/libraries/Address.
    ↪ sol#173-175) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (contracts/libraries/
    ↪ Address.sol#183-193) is never used and should be removed
Address.isContract(address) (contracts/libraries/Address.sol#23-34) is
    ↪ never used and should be removed
Address.sendValue(address,uint256) (contracts/libraries/Address.sol
    ↪ #52-58) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (contracts/libraries/
    ↪ SafeERC20.sol#35-45) is never used and should be removed
SafeERC20.safeTransferETH(address,uint256) (contracts/libraries/
    ↪ SafeERC20.sol#47-51) is never used and should be removed
```

```
SafeMath.add(uint256,uint256) (contracts/libraries/SafeMath.sol#6-11) is
    ↪  never used and should be removed
SafeMath.sqrrt(uint256) (contracts/libraries/SafeMath.sol#56-67) is
    ↪ never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/libraries/SafeMath.sol#13-15)
    ↪ is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/libraries/SafeMath.sol
    ↪ #17-26) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Low level call in Address.sendValue(address,uint256) (contracts/
    ↪ libraries/Address.sol#52-58):
        - (success) = recipient.call{value: amount}() (contracts/
            ↪ libraries/Address.sol#56)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
    ↪ string) (contracts/libraries/Address.sol#125-137):
        - (success,returndata) = target.call{value: value}(data) (
            ↪ contracts/libraries/Address.sol#135)
Low level call in Address._functionCallWithValue(address,bytes,uint256,
    ↪ string) (contracts/libraries/Address.sol#139-165):
        - (success,returndata) = target.call{value: weiValue}(data) (
            ↪ contracts/libraries/Address.sol#148)
Low level call in Address.functionStaticCall(address,bytes,string) (
    ↪ contracts/libraries/Address.sol#183-193):
        - (success,returndata) = target.staticcall(data) (contracts/
            ↪ libraries/Address.sol#191)
Low level call in Address.functionDelegateCall(address,bytes,string) (
    ↪ contracts/libraries/Address.sol#211-221):
        - (success,returndata) = target.delegatecall(data) (contracts/
            ↪ libraries/Address.sol#219)
Low level call in SafeERC20.safeTransferFrom(IERC20,address,address,
    ↪ uint256) (contracts/libraries/SafeERC20.sol#10-21):
        - (success,data) = address(token).call(abi.encodeWithSelector(
          ↪ IERC20.transferFrom.selector,from,to,amount)) (contracts/
          ↪ libraries/SafeERC20.sol#16-18)
Low level call in SafeERC20.safeTransfer(IERC20,address,uint256) (
    ↪ contracts/libraries/SafeERC20.sol#23-33):
        - (success,data) = address(token).call(abi.encodeWithSelector(
          ↪ IERC20.transfer.selector,to,amount)) (contracts/libraries/
          ↪ SafeERC20.sol#28-30)
Low level call in SafeERC20.safeApprove(IERC20,address,uint256) (
    ↪ contracts/libraries/SafeERC20.sol#35-45):
        - (success,data) = address(token).call(abi.encodeWithSelector(
          ↪ IERC20.approve.selector,to,amount)) (contracts/libraries/
          ↪ SafeERC20.sol#40-42)
Low level call in SafeERC20.safeTransferETH(address,uint256) (contracts/
    ↪ libraries/SafeERC20.sol#47-51):
        - (success) = to.call{value: amount}(new bytes(0)) (contracts/
          ↪ libraries/SafeERC20.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Parameter CustomTreasury.deposit(address,uint256,uint256).
    ↪ _principleTokenAddress (contracts/custom/CustomTreasury.sol#48)
    ↪ is not in mixedCase
Parameter CustomTreasury.deposit(address,uint256,uint256).
    ↪ _amountPrincipleToken (contracts/custom/CustomTreasury.sol#48) is
    ↪  not in mixedCase
Parameter CustomTreasury.deposit(address,uint256,uint256).
    ↪ _amountPayoutToken (contracts/custom/CustomTreasury.sol#48) is
    ↪ not in mixedCase
Parameter CustomTreasury.valueOfToken(address,uint256).
    ↪ _principleTokenAddress (contracts/custom/CustomTreasury.sol#62)
    ↪ is not in mixedCase
Parameter CustomTreasury.valueOfToken(address,uint256)._amount (
    ↪ contracts/custom/CustomTreasury.sol#62) is not in mixedCase
```

```
Parameter CustomTreasury.withdraw(address,address,uint256)._token (
    ↪ contracts/custom/CustomTreasury.sol#76) is not in mixedCase
Parameter CustomTreasury.withdraw(address,address,uint256)._destination
    ↪ (contracts/custom/CustomTreasury.sol#76) is not in mixedCase
Parameter CustomTreasury.withdraw(address,address,uint256)._amount (
    ↪ contracts/custom/CustomTreasury.sol#76) is not in mixedCase
Parameter CustomTreasury.toggleBondContract(address)._bondContract (
    ↪ contracts/custom/CustomTreasury.sol#86) is not in mixedCase
Parameter OlympusAccessControlled.setAuthority(IOlympusAuthority).
    ↪ _newAuthority (contracts/types/OlympusAccessControlled.sol#48) is
    ↪  not in mixedCase
Variable OlympusAccessControlled.UNAUTHORIZED (contracts/types/
    ↪ OlympusAccessControlled.sol#11) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

OlympusAccessControlled.UNAUTHORIZED (contracts/types/
    ↪ OlympusAccessControlled.sol#11) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #state-variables-that-could-be-declared-constant

valueOfToken(address,uint256) should be declared external:
        - CustomTreasury.valueOfToken(address,uint256) (contracts/custom/
            ↪ CustomTreasury.sol#62-65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external
contracts/custom/CustomTreasury.sol analyzed (8 contracts with 78
    ↪ detectors), 45 result(s) found
```

Conclusion:

Most of the vulnerabilities found by the analysis have already been addressed by the smart contract code review.

6 Conclusion

In this audit, we examined the design and implementation of NAOS Finance contract and discovered several issues of varying severity. NAOS Finance team addressed 4 issues raised in the initial report and implemented the necessary fixes, while classifying the rest as a risk with low-probability of occurrence. Shellboxes' auditors advised NAOS Finance Team to maintain a high level of vigilance and to keep those findings in mind in order to avoid any future complications.

SHELLBOXES

For a Contract Audit, contact us at contact@shellboxes.com