// HALBORN

Xfai - DEX

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **May 22nd, 2023 - June 19th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	06/15/2023	Luis Buendia
0.2	Document Edit	06/18/2023	Luis Buendia
0.3	Document Edit	06/19/2023	Luis Buendia
0.4	Draft Review	06/19/2023	Gokberk Gulgun
0.5	Draft Review	06/19/2023	Gabi Urrutia
1.0	Remediation Plan	06/27/2023	Luis Buendia
1.1	Remediation Plan Review	06/27/2023	Gokberk Gulgun
1.2	Remediation Plan Review	06/28/2023	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com
Luis Buendia	Halborn	Luis.Buendia@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Xfai engaged Halborn to conduct a security audit on their smart contracts beginning on May 22nd, 2023 and ending on June 19th, 2023. The security assessment was scoped to the smart contracts provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided two weeks for the engagement and assigned a full-time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended.
- Identify potential security issues with the smart contracts.

In summary, Halborn identified some security risks that were addressed and acknowledged by the Xfai team.

1.3 SCOPE

IN-SCOPE:

The security assessment was scoped to the following Xfai Repository :

- InfinityNFTPeriphery.sol
- XfaiFactory.sol
- XfaiINFT.sol
- XfaiPool.sol
- XfaiV0Core.sol
- XfaiV0Periphery01.sol
- xfETH.sol
- TransferHelper.sol
- XfaiLibrary.sol

Xfai Smart Contracts Commit ID:

eb9a7a821ef71e7ad65abe815567d16dfe9d997a

REMEDIATION PLAN:

Xfai Smart Contracts Remediation Commit ID:

3ceca61a67a245e4bb7d7774cfbb34e3eec1aeaa

1.4 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the contracts' solidity code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Smart contract manual code review and walkthrough.
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Manual testing with custom scripts. (Foundry).
- Static Analysis of security for scoped contract, and imported functions manually.
- Testnet deployment (Anvil).

2. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

2.1 EXPLOITABILITY

Attack Origin (AO):

Captures whether the attack requires compromising a specific account.

Attack Cost (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

Attack Complexity (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

Metrics:

Exploitability Metric (m_E)	Metric Value	Numerical Value
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

2.2 IMPACT

Confidentiality (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

Integrity (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

Availability (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

Deposit (D):

Measures the impact to the deposits made to the contract by either users or owners.

Yield (Y):

Measures the impact to the yield generated by the contract for either users or owners.

Metrics:

Impact Metric (m_I)	Metric Value	Numerical Value
Confidentiality (C)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Integrity (I)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Availability (A)	None (A:N)	0
	Low (A:L)	0.25
	Medium (A:M)	0.5
	High (A:H)	0.75
	Critical	1
Deposit (D)	None (D:N)	0
	Low (D:L)	0.25
	Medium (D:M)	0.5
	High (D:H)	0.75
	Critical (D:C)	1
Yield (Y)	None (Y:N)	0
	Low (Y:L)	0.25
	Medium: (Y:M)	0.5
	High: (Y:H)	0.75
	Critical (Y:H)	1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

2.3 SEVERITY COEFFICIENT

Reversibility (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

Scope (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

Coefficient (C)	Coefficient Value	Numerical Value
Reversibility (r)	None (R:N)	1
	Partial (R:P)	0.5
	Full (R:F)	0.25
Scope (s)	Changed (S:C)	1.25
	Unchanged (S:U)	1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

Severity	Score Value Range
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4
Informational	0 - 1.9

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	1	0	13

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
ADDLIQUIDITYETH MISHANDLES DEPOSIT	Medium (5.6)	SOLVED - 06/22/2023
IMPROVEMENTS FOR FLASH CALLS	Informational (0.0)	ACKNOWLEDGED
ENFORCE XFETH CONSTRUCTOR TO RECEIVE ETH	Informational (0.0)	ACKNOWLEDGED
NO SLIPPAGE CONTROL WHEN MINTING XFETH	Informational (0.0)	ACKNOWLEDGED
ADDLIQUIDITYETH FUNCTION MAY REVERT ON FIRST DEPOSIT	Informational (0.0)	ACKNOWLEDGED
ABSENCE OF TOKEN OWNERSHIP CHECK IN THE BOOST FUNCTION	Informational (0.0)	ACKNOWLEDGED
THE PROTOCOL DOES NOT ALLOW TO ADD LIQUIDITY USING XFETH	Informational (0.0)	FUTURE RELEASE
REMOVING LIQUIDITY CAN REVERT IF TOKEN ORDER IS NOT SET	Informational (0.0)	ACKNOWLEDGED
REDUNDANT VARIABLE	Informational (0.0)	SOLVED - 06/22/2023
LACK OF UPGRADABILITY PATTERN	Informational (0.0)	ACKNOWLEDGED
CONVERT STRINGS FOR CUSTOM ERRORS TO SAVE GAS	Informational (0.0)	ACKNOWLEDGED
FLOATING PRAGMA	Informational (0.0)	ACKNOWLEDGED
LACK OF PAUSE FUNCTIONALITY ON THE CONTRACTS	Informational (0.0)	ACKNOWLEDGED
LACK OF TWO STEP OWNERSHIP TRANSFER	Informational (0.0)	ACKNOWLEDGED

FINDINGS & TECH DETAILS

4.1 (HAL-01) ADDLIQUIDITYETH MISHANDLES DEPOSIT - MEDIUM (5.6)

Description:

The function addLiquidityETH from the XfaiV0Periphery01.sol contract takes ether received from the user and deposits it into the WETH/XFETH pool. If the pool has liquidity, it computes the amount to deposit of each asset. First, it calculates the proportional amount of ETH sent for the sum of the amounts of each token that the pool has. Then it subtracts the obtained value to the amount of sent ether to transform it to xfETH.

As xfETH tends to increase its value, this calculation can result in a non-optimal deposit, making users obtain less LP tokens than the optimal distribution may achieve.

Code Location:

Listing 1: xfETH.sol

```
function addLiquidityETH(
address _to,
uint _deadline
) external payable override ensure(_deadline) returns (uint
↳ liquidity) {
    address _weth = weth; // gas saving
    uint amountETH;
    uint amountXfETHtoETH;
    address pool = IXfaiFactory(factory).getPool(_weth);
    if (pool == address(0)) {
      // create the pool if it doesn't exist yet
      pool = IXfaiFactory(factory).createPool(_weth);
    }
    (uint ETHReserve, uint xfETHReserve) = IXfaiPool(pool).
↳ getStates();
    if (ETHReserve == 0 && xfETHReserve == 0) {
      (amountETH, amountXfETHtoETH) = (msg.value / 2, msg.value /
↳ 2);
    } else {
      amountETH = (msg.value * ETHReserve) / (ETHReserve +
↳ xfETHReserve);
      amountXfETHtoETH = msg.value - amountETH;
    }
    uint amountXfETH = IXFETH(xfETH).deposit{value:
↳ amountXfETHtoETH}();
    IWETH(_weth).deposit{value: amountETH}();
    TransferHelper.safeTransfer(xfETH, pool, amountXfETH);
    TransferHelper.safeTransfer(_weth, pool, amountETH);
    liquidity = IXfaiV0Core(core).mint(_weth, _to);
    require(msg.value == amountETH + amountXfETHtoETH, '
↳ XfaiV0Periphery01: INSUFFICIENT_AMOUNT');
}
```

Proof Of Concept:

The issue resides in the fact that the calculation for the corresponding amount of xfETH is the amount of ETH that will be converted to xfETH. Thus, if the value of xfETH is higher than ETH, this amount will decrease when converted to the token system. As the formula for minting LP tokens takes the minimum of the resulting product of the added tokens, the LP tokens minted may be lower than other distributions.

The next scenario is used to illustrate the described issue.

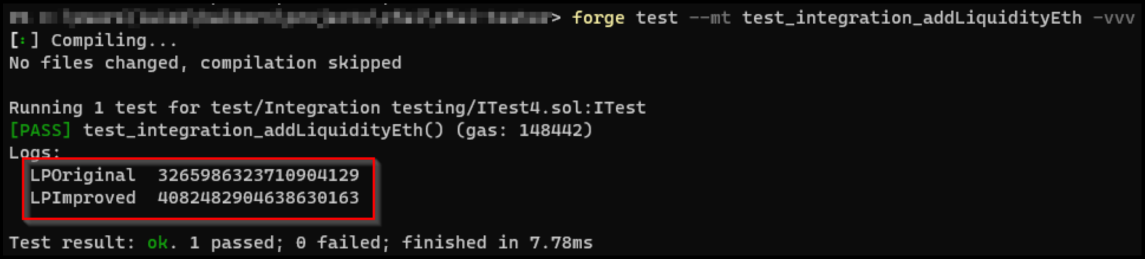
1. ETH to Xfeth price has a relation of 3 Eth to 2 xfETH.
2. ETH balance of xfETH contract is 1.5 bigger than the total supply.
3. The WETH/XFETH pool is balanced, with 300 WETH on reserve and 200 xfETH on weight.
4. User deposits 10 ether.
5. The test also computes the LP minted with a different distribution based on the values previous to the original deposit.

Listing 2: ITest4.sol

```
pragma solidity ^0.8.19;

import 'test/Deployer.sol';

```

```solidity
contract ITest is Deployer {

    MockWETH weth2;

    function setUp() public override {
        super.setUp();

        deal(address(this), 1000 ether);
        //xfaiperiphery.addLiquidityETH{value: 20 ether}(address(this), block.timestamp+1000);
        xfactory.createPool(address(weth));
        address pool = xfactory.getPool(address(weth));
        xfeth.deposit{value: 20 ether}();
        weth.deposit{value: 30 ether}();
        xfeth.transfer(pool, 20 ether);
        weth.transfer(pool, 30 ether);
        xfaicore.mint(address(weth), address(this));
    }

    function test_integration_addLiquidityEth() public {

        uint256 newBalance = (address(xfeth).balance * 3) / 2;
        deal(address(xfeth), newBalance);

        address pool = IXfaiFactory(xfactory).getPool(address(weth));
        (uint ETHReserve, uint xfETHReserve) = IXfaiPool(pool).getStates();

        uint256 totalSupply = MockERC20(pool).totalSupply();

        uint256 originalLP = xfaiperiphery.addLiquidityETH{value: 10 ether}(address(this), block.timestamp+1000);

        uint amountETHNew = 5 ether;
        uint amountXfETHtoETHNew = 10 ether - amountETHNew;
        uint amountXfETHNew = xfeth.ETHToXfETH(amountXfETHtoETHNew);

        uint liquidityNew = Math.min((amountETHNew * totalSupply) / ETHReserve, (amountXfETHNew * totalSupply) / xfETHReserve);

        console.log('LPOriginal ', originalLP);
        console.log('LPImproved ', liquidityNew);
    }
}
```

The next screenshots show the difference between the LP tokens obtained.



```
[redacted]> forge test --mt test_integration_addLiquidityEth -vvv
[:] Compiling...
No files changed, compilation skipped

Running 1 test for test/Integration testing/ITest4.sol:ITest
[PASS] test_integration_addLiquidityEth() (gas: 148442)
Logs:
  LPOriginal  3265986323710904129
  LPImproved  4082482904638630163

Test result: ok. 1 passed; 0 failed; finished in 7.78ms
```

As it is possible to observe, the difference does exist.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:M/Y:L/R:N/S:U (5.6)

Recommendation:

Consider using a new formula that obtains the optimal distribution. As this may not be an easy task, it can also be a reasonable approach to establish a user defined minimal distribution to perform the calculations. It is also advised to set a minimal LP tokens to be obtained to avoid front-running issues.

Remediation Plan:

SOLVED: The Xfai team solved the issue on the next commit ID 3ceca61a67a245e4bb7d7774cfbb34e3eec1aeaa.

4.2 (HAL-02) IMPROVEMENTS FOR FLASH CALLS - INFORMATIONAL (0.0)

Description:

The Xfai Protocol has two different functions to perform flash calls. First one through a flash mint on the Xfeth.sol contract. The second one is a classic flash loan on the XfaiV0Core.sol contract.

Although the functions are technically correct from a basic functionality point of view, the standard established on EIP-3165 is not fulfilled.

The standard improvements are:

- Implement a flashFee view function to compute the fee for a given token amount.
- Implement a maxFlashLoan view function to obtain the maximum number of tokens available.
- Return true if the execution is successful.
- Control the return value of the callback function.
- Send the parameters of msg.sender, token, amount, fee and data as inputs to the receiver callback function.

Code Location:

Listing 3: XFETH.sol (Line 186)

```
function flashMint(uint _amount) external override nonReentrant
 ↳ isPublic {
    // get current ETH balance
    uint ETHBalance = address(this).balance;
    uint xfETHTotalSupply = totalSupply();

    // compute fee
    uint fee = (_amount * flashMintFee) / 10000;

    // mint tokens
    _mint(msg.sender, _amount);
184
185     // hand control to borrower
186     IBorrower(msg.sender).executeOnFlashMint(_amount);
187
188     // burn tokens + fee
189     _burn(msg.sender, _amount + fee); // reverts if `msg.sender` does not have enough tokens to burn
190
191     // double-check that the contract's ETH balance has not decreased
192     assert(address(this).balance >= ETHBalance);
193
194     // double-check that the contract's xfETH supply has decreased
195     assert(totalSupply() < xfETHTotalSupply);
196
197     emit FlashMint(msg.sender, _amount);
198 }
```

Listing 4: XfaiV0Core.sol (Line 335)

```
323 function flashLoan(
324 address _token,
325 uint _amount,
326 address _to,
327 bytes calldata _data
328 ) external override pausable singleLock(_token) {
329     require(_to != address(0), 'XfaiV0Core INVALID_TO');
330     address pool = XfaiLibrary.poolFor(_token, factory, poolCodeHash);
331     (uint reserve, ) = IXfaiPool(pool).getStates();
332     require(_amount <= reserve, 'XfaiV0Core: INSUFFICIENT_OUTPUT_AMOUNT');
333     uint balance = IERC20(_token).balanceOf(pool);
334     IXfaiPool(pool).linkedTransfer(_token, _to, _amount); // optimistically transfer tokens
335     IXfaiV0FlashLoan(_to).flashLoan(pool, _amount, _data);
336     require(
337       IERC20(_token).balanceOf(pool) >= balance + ((_amount * getTotalFee()) / 10000),
338       'XfaiV0Core: INSUFFICIENT_AMOUNT_RETURNED'
339     );
340     IXfaiPool(pool).linkedTransfer(_token, infinityNFT, (_amount * infinityNFTFee) / 10000); // send lnft fee to fee collecting contract
```

```
    IXfaiPool(pool).update(IERC20(_token).balanceOf(pool), IERC20(
 ↳ xfETH).balanceOf(pool));
    emit FlashLoan(_to, _amount);
}
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

From all the previous detailed improvements, consider implementing all of them. Nonetheless, there is one that the Halborn team strongly suggests. These are the parameters specified for the receive callback function.

By adding the msg.sender it is ensured that the receiver fallback function executes from trusted origins. Adding the fee avoids the receiver contract to perform any computation or further contract calls to obtain this value. Sending the token address simplifies the logic of the receiver. Finally, the data is absolutely needed to execute the adequate control flow statements on the receiver side.

Moreover, consider implementing interfaces following the standard to simplify the usability of the protocol.

Reference EIP-3156

Remediation Plan:

ACKNOWLEDGED: The Xfai team acknowledged this finding.

4.3 (HAL-03) ENFORCE XFETH CONSTRUCTOR TO RECEIVE ETH - INFORMATIONAL (0.0)

Description:

The xfEth.solcontract must receive ether on the constructor. The formula used for minting liquidity on the deposit function uses the total supply of xfeth on the numerator of a division to calculate the number of tokens to give in return. If this value is zero, the returned amount will always be zero.

Code Location:

Listing 5: XFETH.sol

```
constructor(address _owner, uint _flashMintFee) payable ERC20() {
    _mint(address(0), msg.value);
    owner = _owner;
    flashMintFee = _flashMintFee;
    _status = _NOT_ENTERED;
    _name = 'Xfai ETH';
    _symbol = 'XFETH';
}
```

Listing 6: XFETH.sol

```
function deposit() public payable override nonReentrant returns (
↳ uint amountInXfETH) {
    amountInXfETH = (msg.value * totalSupply()) / (address(this).
↳ balance - msg.value);
    _mint(msg.sender, amountInXfETH);
    emit Deposit(msg.sender, amountInXfETH, msg.value);
}
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

Consider adding a require statement to enforce the contract to receive ether when deployed.

Remediation Plan:

ACKNOWLEDGED: The Xfai team acknowledged this finding.

4.4 (HAL-04) NO SLIPPAGE CONTROL WHEN MINTING XFETH - INFORMATIONAL (0.0)

Description:

The deposit function of xfETH.sol contract allows sending ether and receive xfeth token in return. The nature of this contract allows xfeth value to increase, by reducing the total supply through the fee burned on the flashMint function and remaining the same native token balance.

The Xfai DEX is designed to work in optimal conditions, with flashMints being constantly used to take advantage of arbitrage opportunities. It also means, any time an arbitrageur succeeds, all the pools will get unbalanced, starting a virtuous loop of constant profit for all actors in the system.

Due to this, it is plausible that a user who attempts to deposit ETH and get Xfeth in return does not obtain the desired amount. Moreover, in certain value ranges, it is possible to send ETH and obtain zero Xfeth in return. So, considering these two scenarios, it is sensible to consider implementing a slippage control of the minimal amount of Xfeth expected on the deposit function.

Code Location:

Listing 7: xfETH.sol

```
function deposit() public payable override nonReentrant returns (
↳ uint amountInXfETH) {
    amountInXfETH = (msg.value * totalSupply()) / (address(this).
↳ balance - msg.value);
    _mint(msg.sender, amountInXfETH);
    emit Deposit(msg.sender, amountInXfETH, msg.value);
}
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

Consider adding a parameter variable for the deposit function that allows the user to revert the transaction if the Xfeth returned value is lower than expected.

Remediation Plan:

ACKNOWLEDGED: The Xfai team acknowledged this finding.

4.5 (HAL-05) ADDLIQUIDITYETH FUNCTION MAY REVERT ON FIRST DEPOSIT - INFORMATIONAL (0.0)

Description:

The addLiquidityETH function from the XfaiV0Periphery01.sol contract, on the first deposit to the Weth/Xfeth pool, splits the value of ETH sent by the user to use each amount to obtain weth and xfeth. With those obtained amounts at the end of the function, it performs a require statement, where those values are added and need to be equal to the msg.value.

The problem of this implementation arises when the first deposit is done with and odd ETH number. In this case, the require statement will revert due to solidity precision loss.

Code Location:

Listing 8: XfaiV0Periphery01.sol (Lines 177,187)

```
163 function addLiquidityETH(
164 address _to,
165 uint _deadline
166 ) external payable override ensure(_deadline) returns (uint
 ↳ liquidity) {
167     address _weth = weth; // gas saving
168     uint amountETH;
169     uint amountXfETHtoETH;
170     address pool = IXfaiFactory(factory).getPool(_weth);
171     if (pool == address(0)) {
172       // create the pool if it doesn't exist yet
173       pool = IXfaiFactory(factory).createPool(_weth);
174     }
175     (uint ETHReserve, uint xfETHReserve) = IXfaiPool(pool).
 ↳ getStates();
176     if (ETHReserve == 0 && xfETHReserve == 0) {
177       (amountETH, amountXfETHtoETH) = (msg.value / 2, msg.value /
 ↳ 2);
```

```
    } else {
        amountETH = (msg.value * ETHReserve) / (ETHReserve +
↳ xfETHReserve);
        amountXfETHtoETH = msg.value - amountETH;
    }
    uint amountXfETH = IXFETH(xfETH).deposit{value:
↳ amountXfETHtoETH}();
    IWETH(_weth).deposit{value: amountETH}();
    TransferHelper.safeTransfer(xfETH, pool, amountXfETH);
    TransferHelper.safeTransfer(_weth, pool, amountETH);
    liquidity = IXfaiV0Core(core).mint(_weth, _to);
    require(msg.value == amountETH + amountXfETHtoETH, '
↳ XfaiV0Periphery01: INSUFFICIENT_AMOUNT');
}
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

The case of doing a first deposit with an odd number is an edge case. Nevertheless, it is possible to implement a solution for avoiding the possibility to revert on a fair deposit.

Remediation Plan:

ACKNOWLEDGED: The Xfai team acknowledged this finding.

4.6 (HAL-06) ABSENCE OF TOKEN OWNERSHIP CHECK IN THE BOOST FUNCTION - INFORMATIONAL (0.0)

Description:

The function boost of the XfaiINFT.sol contract allows increasing the number of shares of a specific NFT token ID, based on the amount of XFIT tokens that the factory has received.

However, the function does not check if the caller owns the indicated token ID. This does not represent a security risk, but it can prevent certain unwanted scenarios from the user perspective.

Code Location:

Listing 9: XfaiINFT.sol (Line 217)

```
211 function boost(uint _tokenID) external override lock returns (uint
  ↳ share) {
212     require(_tokenID <= counter, 'XfaiINFT: Inexistent_ID');
213     uint amount = IERC20(underlyingToken).balanceOf(factory) -
  ↳ reserve;
214     require(amount != 0, 'XfaiINFT: INSUFICIENT_AMOUNT');
215     reserve += amount;
216     share = (1e18 * amount) / (reserve + initialReserve);
217     INFTShares[_tokenID] += share;
218     totalSharesIssued += share;
219     emit Boost(msg.sender, share, _tokenID);
220 }
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

Consider adding a require statement that prevents any user except the owner to increase the shares of the specified NFT.

Remediation Plan:

ACKNOWLEDGED: The Xfai team acknowledged this finding.

4.7 (HAL-07) THE PROTOCOL DOES NOT ALLOW TO ADD LIQUIDITY USING XFETH - INFORMATIONAL (0.0)

Description:

The current implementation of Xfai Protocol does not allow to users to provide liquidity using Xfeth. The current functions of the protocol force the user to supply ETH that is transformed into Xfeth.

This can be an issue for the users, specially to the ones that already have xfeth minted, in the case of an appreciation of the xfeth token to ETH.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

Consider adding on the XfaiV0Periphery01.sol contract the necessary functions to allow adding liquidity with the main token of the protocol.

Remediation Plan:

PENDING: The Xfai team plans to implement this functionality in the next release.

4.8 (HAL-08) REMOVING LIQUIDITY CAN REVERT IF TOKEN ORDER IS NOT SET - INFORMATIONAL (0.0)

Description:

The internal _removeLiquidity function of the XfaiV0Periphery01.sol contract, receives two token addresses as parameters. These inputs come from the external function removeLiquidity. If none of the inputs tokens address is the WETH contract, the internal function is called, passing directly the input user parameters received. The function then attempts to retrieve the address of the pool corresponding to the token in the first place, however, if the address corresponds to the Xfeth token, the transaction reverts, attempting to call a function of the zero address.

Code Location:

Listing 10: XfaiV0Periphery01.sol (Lines 200,201)

```
191 function _removeLiquidity(
192 address _token0,
193 address _token1,
194 uint _liquidity,
195 uint _amount0Min,
196 uint _amount1Min,
197 address _to
198 ) private returns (uint amount0, uint amount1) {
199     address _core = core; // gas saving
200     address pool = XfaiLibrary.poolFor(_token0, factory,
  ↳ poolCodeHash);
201     TransferHelper.safeTransferFrom(pool, msg.sender, _core,
  ↳ _liquidity);
202     (amount0, amount1) = IXfaiV0Core(_core).burn(_token0, _token1,
  ↳  _to);
203     require(amount0 >= _amount0Min, 'XfaiV0Periphery01:
  ↳ INSUFFICIENT_AMOUNT0');
204     require(amount1 >= _amount1Min, 'XfaiV0Periphery01:
  ↳ INSUFFICIENT_AMOUNT1');
205 }
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

Consider controlling the tokens address to avoid unnecessary reverts.

Remediation Plan:

ACKNOWLEDGED: The Xfai team acknowledged this finding.

4.9 (HAL-09) REDUNDANT VARIABLE - INFORMATIONAL (0.0)

Description:

Line 363 of the XfaiV0Periphery01.sol contract is not required, as the stored value of the variable is not used.

Code Location:

Listing 11: XfaiV0Periphery01.sol (Line 363)

```
346 function swapTokensForExactTokens(
347 address _to,
348 address _token0,
349 address _token1,
350 uint _amount1Out,
351 uint _amount0InMax,
352 uint _deadline
353 ) external override ensure(_deadline) returns (uint amount0In) {
354     address pool0;
355     address pool1;
356     if (_token0 == xfETH) {
357       pool0 = XfaiLibrary.poolFor(_token1, factory, poolCodeHash);
358       pool1 = XfaiLibrary.poolFor(_token1, factory, poolCodeHash);
359       (uint r, uint w) = IXfaiPool(pool0).getStates();
360       amount0In = XfaiLibrary.getAmountIn(w, r, _amount1Out,
    ↳ IXfaiV0Core(core).getTotalFee());
361     } else if (_token1 == xfETH) {
362       pool0 = XfaiLibrary.poolFor(_token0, factory, poolCodeHash);
363       pool1 = XfaiLibrary.poolFor(_token0, factory, poolCodeHash);
364       (uint r, uint w) = IXfaiPool(pool0).getStates();
365       amount0In = XfaiLibrary.getAmountIn(r, w, _amount1Out,
    ↳ IXfaiV0Core(core).getTotalFee());
366     } else {
367       pool0 = XfaiLibrary.poolFor(_token0, factory, poolCodeHash);
368       pool1 = XfaiLibrary.poolFor(_token1, factory, poolCodeHash);
369       amount0In = XfaiLibrary.getAmountsIn(
370         pool0,
371         pool1,
372         _amount1Out,
            IXfaiV0Core(core).getTotalFee()
          );
        }
        require(amount0In <= _amount0InMax, 'XfaiV0Periphery01:
  ↳ INSUFFICIENT_INPUT_AMOUNT');
        TransferHelper.safeTransferFrom(_token0, msg.sender, pool0,
  ↳ amount0In);
        _swap(_token0, _token1, _to);
}
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

Consider erasing the unnecessary lines from the code base.

Remediation Plan:

SOLVED: The Xfai team removed the redundant variable on the next commit ID 3ceca61a67a245e4bb7d7774cfbb34e3eec1aeaa.

4.10 (HAL-10) LACK OF UPGRADABILITY PATTERN - INFORMATIONAL (0.0)

Description:

The current version of the project only allows the core contract to be upgraded. This can be useful either to fix potential unwanted behaviors and also to add new functionalities in future releases of the protocol.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

Consider adding proxy contracts that allow other components of the protocol to be upgradable.

Remediation Plan:

ACKNOWLEDGED: The Xfai team acknowledged this finding.

4.11 (HAL-11) CONVERT STRINGS FOR CUSTOM ERRORS TO SAVE GAS - INFORMATIONAL (0.0)

Description:

Custom errors are available from Solidity version 0.8.4. Custom errors save ~50 gas each time they are hit by avoiding having to allocate and store the revert string. Not defining strings also saves deployment gas.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

Consider replacing all revert strings with custom errors.

Remediation Plan:

ACKNOWLEDGED: The Xfai team acknowledged this finding.

4.12 (HAL-12) FLOATING PRAGMA - INFORMATIONAL (0.0)

Description:

Contracts should be deployed with the same compiler version and flags used during development and testing. Locking the pragma helps to ensure that contracts do not accidentally get deployed using another pragma. For example, an outdated pragma version might introduce bugs that affect the contract system negatively.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

Consider locking the pragma version in the smart contracts. It is not recommended to use a floating pragma in production.

For example: `pragma solidity 0.8.20;`

Remediation Plan:

ACKNOWLEDGED: The Xfai team acknowledged this finding.

4.13 (HAL-13) LACK OF PAUSE FUNCTIONALITY ON THE CONTRACTS - INFORMATIONAL (0.0)

Description:

Although the core contract can be paused, the xfETH.sol contract and the XfaiINFT.sol do not contain any pausable modifier.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

Consider implementing the pausable security model on the contracts left.

Remediation Plan:

ACKNOWLEDGED: The Xfai team acknowledged this finding.

4.14 (HAL-14) LACK OF TWO STEP OWNERSHIP TRANSFER - INFORMATIONAL (0.0)

Description:

The current ownership transfer process for all the contracts inheriting from Ownable involves the current owner calling the transferOwnership() function:

Listing 12: Ownable.sol

```
function transferOwnership(address newOwner) public virtual
↳ onlyOwner {
    require(newOwner != address(0), "Ownable: new owner is the
↳ zero address");
    _setOwner(newOwner);
}
```

If the nominated EOA account is not a valid account, it is entirely possible that the owner may accidentally transfer ownership to an uncontrolled account, losing the access to all functions with the onlyOwner modifier.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

It is recommended to implement a two-step process where the owner nominates an account and the nominated account needs to call an acceptOwnership() function for the transfer of the ownership to fully succeed. This ensures the nominated EOA account is a valid and active account.

Remediation Plan:

ACKNOWLEDGED: The Xfai team acknowledged this finding.

AUTOMATED TESTING

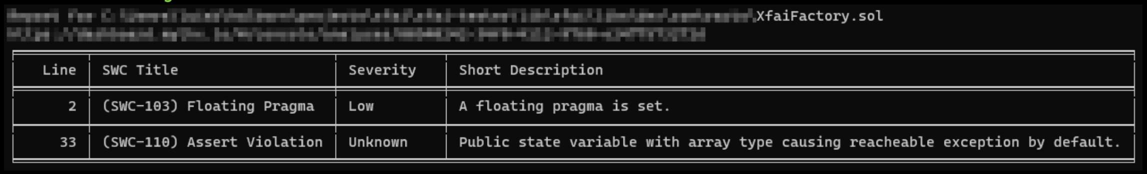
5.1 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the smart contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

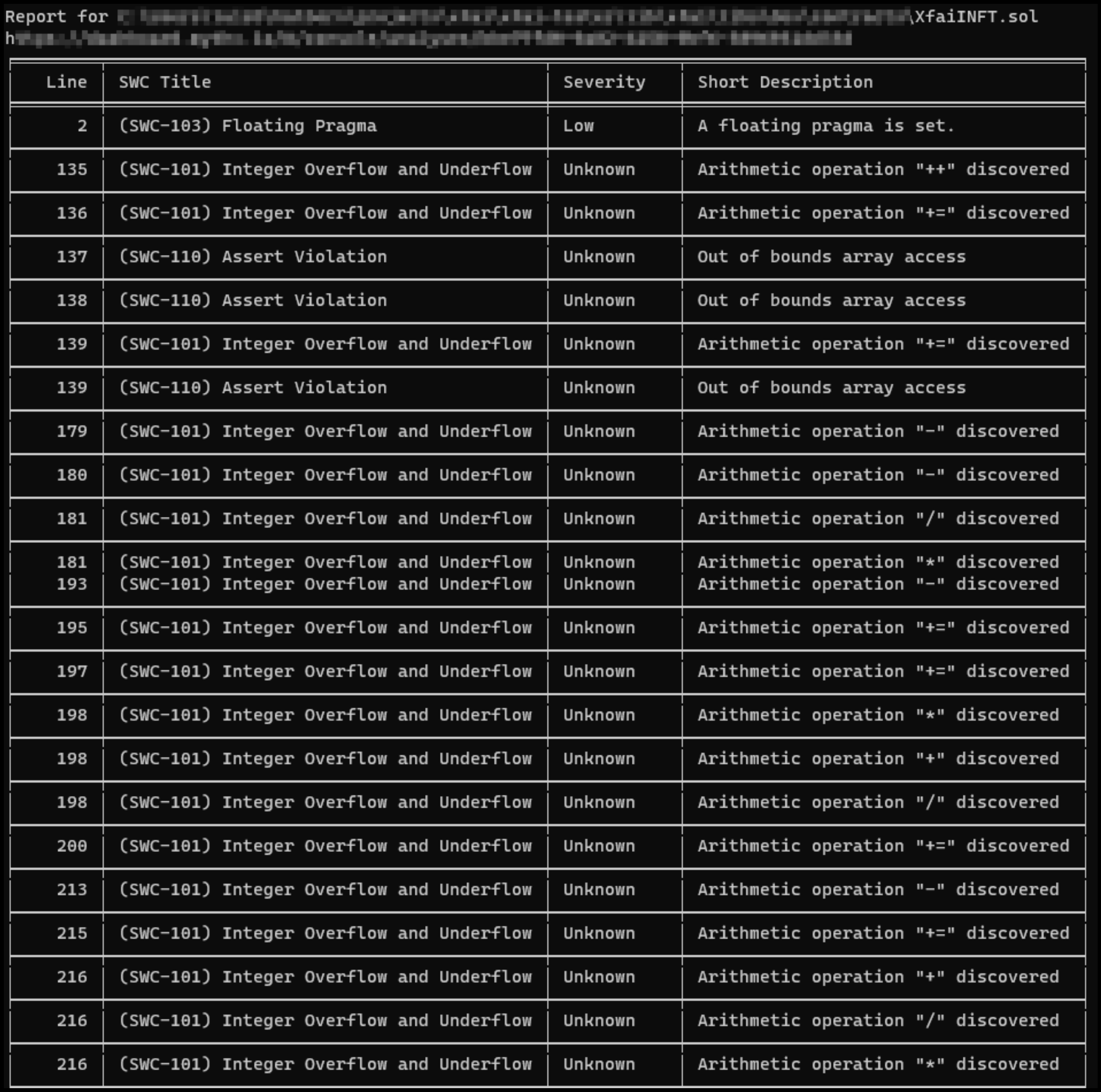
MythX results:

XfaiFactory.sol



```
[redacted]XfaiFactory.sol

╒════════╤════════════════════════════╤════════════╤══════════════════════════════════════════════════════════════════════╕
│   Line │ SWC Title                  │ Severity   │ Short Description                                                    │
╞════════╪════════════════════════════╪════════════╪══════════════════════════════════════════════════════════════════════╡
│      2 │ (SWC-103) Floating Pragma  │ Low        │ A floating pragma is set.                                            │
├────────┼────────────────────────────┼────────────┼──────────────────────────────────────────────────────────────────────┤
│     33 │ (SWC-110) Assert Violation │ Unknown    │ Public state variable with array type causing reacheable exception by default. │
╘════════╧════════════════════════════╧════════════╧══════════════════════════════════════════════════════════════════════╛
```

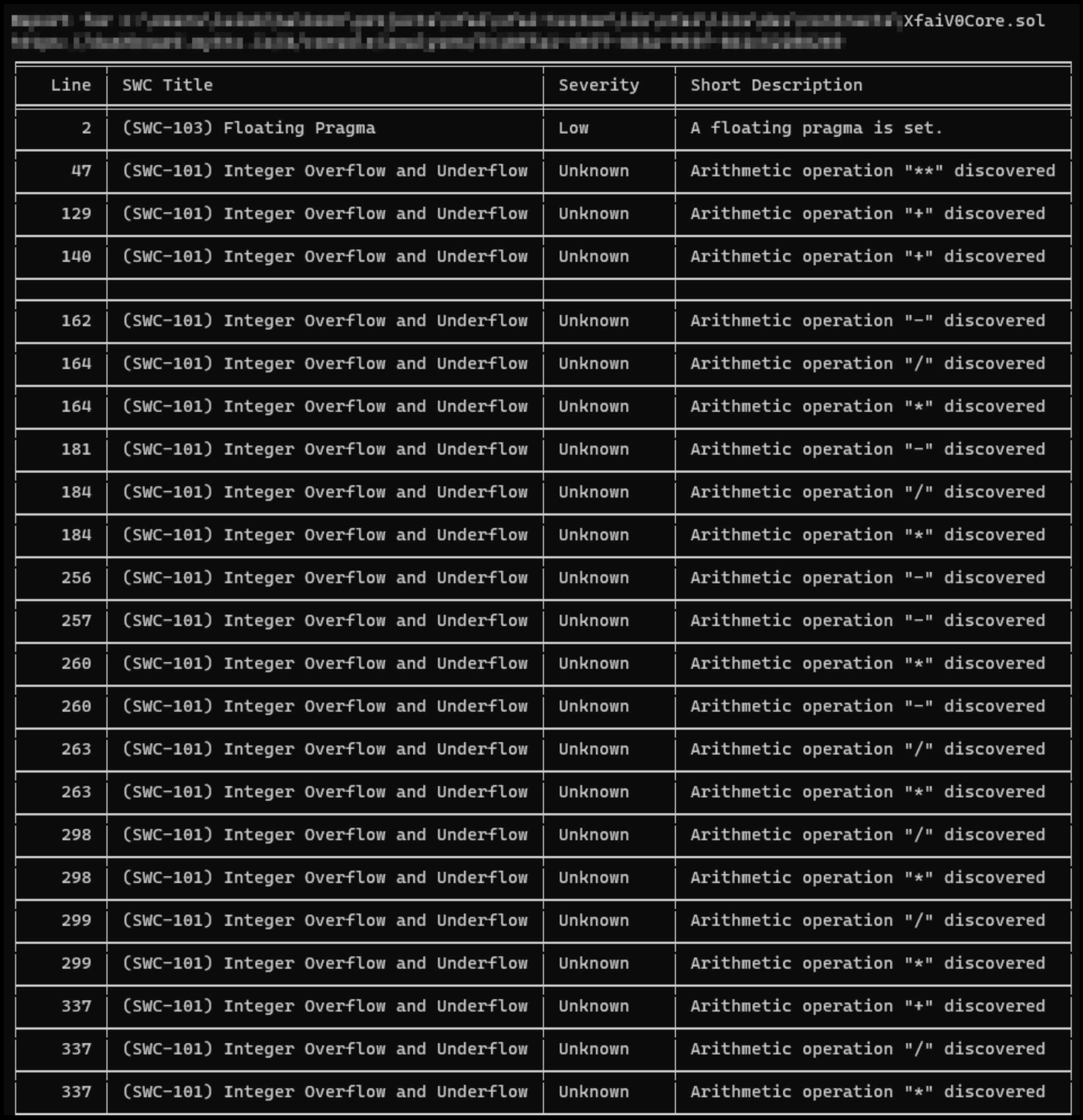
XfaiINFT.sol



```
Report for [redacted]\XfaiINFT.sol
h[redacted]
╒════════╤═══════════════════════════════════════════╤════════════╤═══════════════════════════════════════╕
│  Line  │ SWC Title                                 │ Severity   │ Short Description                     │
╞════════╪═══════════════════════════════════════════╪════════════╪═══════════════════════════════════════╡
│      2 │ (SWC-103) Floating Pragma                 │ Low        │ A floating pragma is set.             │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    135 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "++" discovered  │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    136 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "+=" discovered  │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    137 │ (SWC-110) Assert Violation                │ Unknown    │ Out of bounds array access            │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    138 │ (SWC-110) Assert Violation                │ Unknown    │ Out of bounds array access            │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    139 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "+=" discovered  │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    139 │ (SWC-110) Assert Violation                │ Unknown    │ Out of bounds array access            │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    179 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "-" discovered   │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    180 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "-" discovered   │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    181 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "/" discovered   │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    181 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "*" discovered   │
│    193 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "-" discovered   │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    195 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "+=" discovered  │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    197 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "+=" discovered  │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    198 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "*" discovered   │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    198 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "+" discovered   │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    198 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "/" discovered   │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    200 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "+=" discovered  │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    213 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "-" discovered   │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    215 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "+=" discovered  │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    216 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "+" discovered   │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    216 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "/" discovered   │
├────────┼───────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    216 │ (SWC-101) Integer Overflow and Underflow  │ Unknown    │ Arithmetic operation "*" discovered   │
╘════════╧═══════════════════════════════════════════╧════════════╧═══════════════════════════════════════╛
```

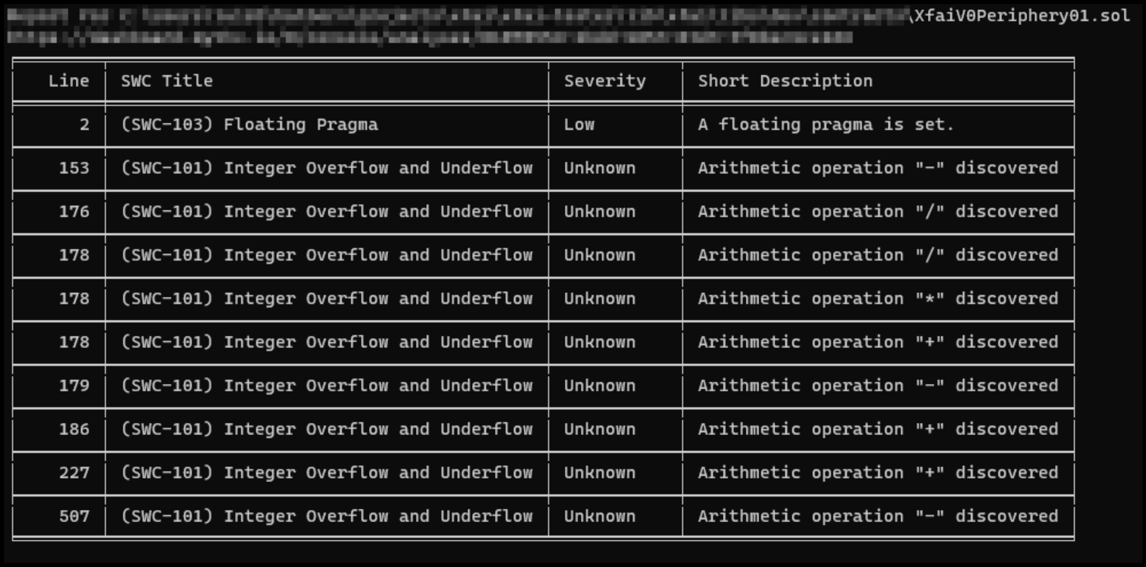
XfaiPool.sol

XfaiV0Core.sol



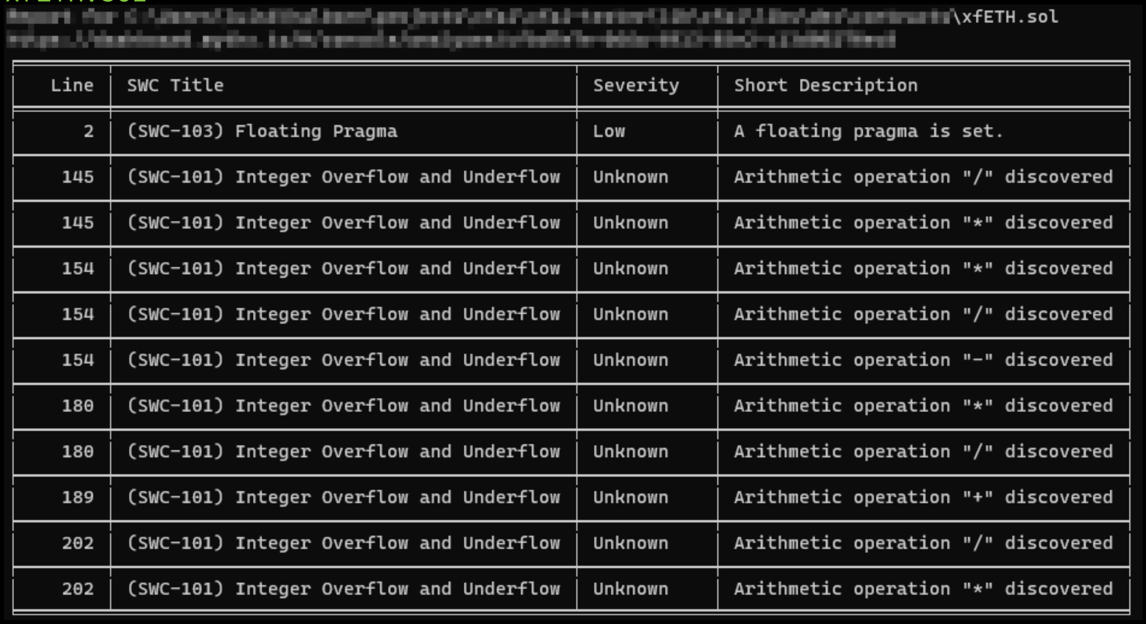
```
[redacted]XfaiV0Core.sol

 Line | SWC Title                                    | Severity | Short Description
------+----------------------------------------------+----------+------------------------------------
    2 | (SWC-103) Floating Pragma                    | Low      | A floating pragma is set.
   47 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "**" discovered
  129 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "+" discovered
  140 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "+" discovered
      |                                              |          |
  162 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "-" discovered
  164 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "/" discovered
  164 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "*" discovered
  181 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "-" discovered
  184 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "/" discovered
  184 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "*" discovered
  256 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "-" discovered
  257 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "-" discovered
  260 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "*" discovered
  260 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "-" discovered
  263 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "/" discovered
  263 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "*" discovered
  298 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "/" discovered
  298 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "*" discovered
  299 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "/" discovered
  299 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "*" discovered
  337 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "+" discovered
  337 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "/" discovered
  337 | (SWC-101) Integer Overflow and Underflow     | Unknown  | Arithmetic operation "*" discovered
```

XfaiV0Periphery01.sol



```
[redacted]\XfaiV0Periphery01.sol
[redacted]

| Line | SWC Title                                | Severity | Short Description                     |
|    2 | (SWC-103) Floating Pragma                | Low      | A floating pragma is set.             |
|  153 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "-" discovered   |
|  176 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "/" discovered   |
|  178 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "/" discovered   |
|  178 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "*" discovered   |
|  178 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "+" discovered   |
|  179 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "-" discovered   |
|  186 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "+" discovered   |
|  227 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "+" discovered   |
|  507 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "-" discovered   |
```

xfETH.sol



```
[redacted]\xfETH.sol
[redacted]

| Line | SWC Title                                | Severity | Short Description                     |
|    2 | (SWC-103) Floating Pragma                | Low      | A floating pragma is set.             |
|  145 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "/" discovered   |
|  145 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "*" discovered   |
|  154 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "*" discovered   |
|  154 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "/" discovered   |
|  154 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "-" discovered   |
|  180 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "*" discovered   |
|  180 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "/" discovered   |
|  189 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "+" discovered   |
|  202 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "/" discovered   |
|  202 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "*" discovered   |
```

- No major issues found by MythX.

THANK YOU FOR CHOOSING

HALBORN