HALBORN

Staderlabs - sFTMX Fantom

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **April 13, 2022 - April 20, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY 3
CONTACTS 3
1 EXECUTIVE OVERVIEW 4
1.1 INTRODUCTION 5
1.2 AUDIT SUMMARY 5
1.3 TEST APPROACH & METHODOLOGY 5
RISK METHODOLOGY 6
1.4 SCOPE 8
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW 9
3 FINDINGS & TECH DETAILS 10
3.1 (HAL-01) OWNER CAN RENOUNCE OWNERSHIP - LOW 12
Description 12
Code Location 12
Risk Level 12
Recommendation 12
Remediation Plan 13
3.2 (HAL-02) MISSING EVENTS FOR ADMIN ONLY FUNCTIONS THAT CHANGE CRITICAL PARAMETERS - INFORMATIONAL 14
Description 14
Code Location 14
Risk Level 14
Recommendation 15
Remediation Plan 15
3.3 (HAL-03) ZERO ADDRESS NOT CHECKED - INFORMATIONAL 16
Description 16

Code Location 16
Risk Level 16
Recommendation 16
Remediation Plan 16
4 AUTOMATED TESTING 16
4.1 STATIC ANALYSIS REPORT 18
Description 18
Slither results 18
4.2 AUTOMATED SECURITY SCAN 22
Description 22
MythX results 22

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	04/18/2022	Omar Alshaeb
0.2	Draft Review	04/20/2022	Gabi Urrutia
1.0	Remediation Plan	04/25/2022	Omar Alshaeb
1.1	Remediation Plan Review	04/25/2022	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
Omar Alshaeb	Halborn	Omar.Alshaeb@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Staderlabs engaged Halborn to conduct a security audit on their smart contracts beginning on April 13, 2022 and ending on April 20, 2022 . The security assessment was scoped to the smart contracts provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided one week 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 by the Staderlabs team.

1.3 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 bridge 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
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

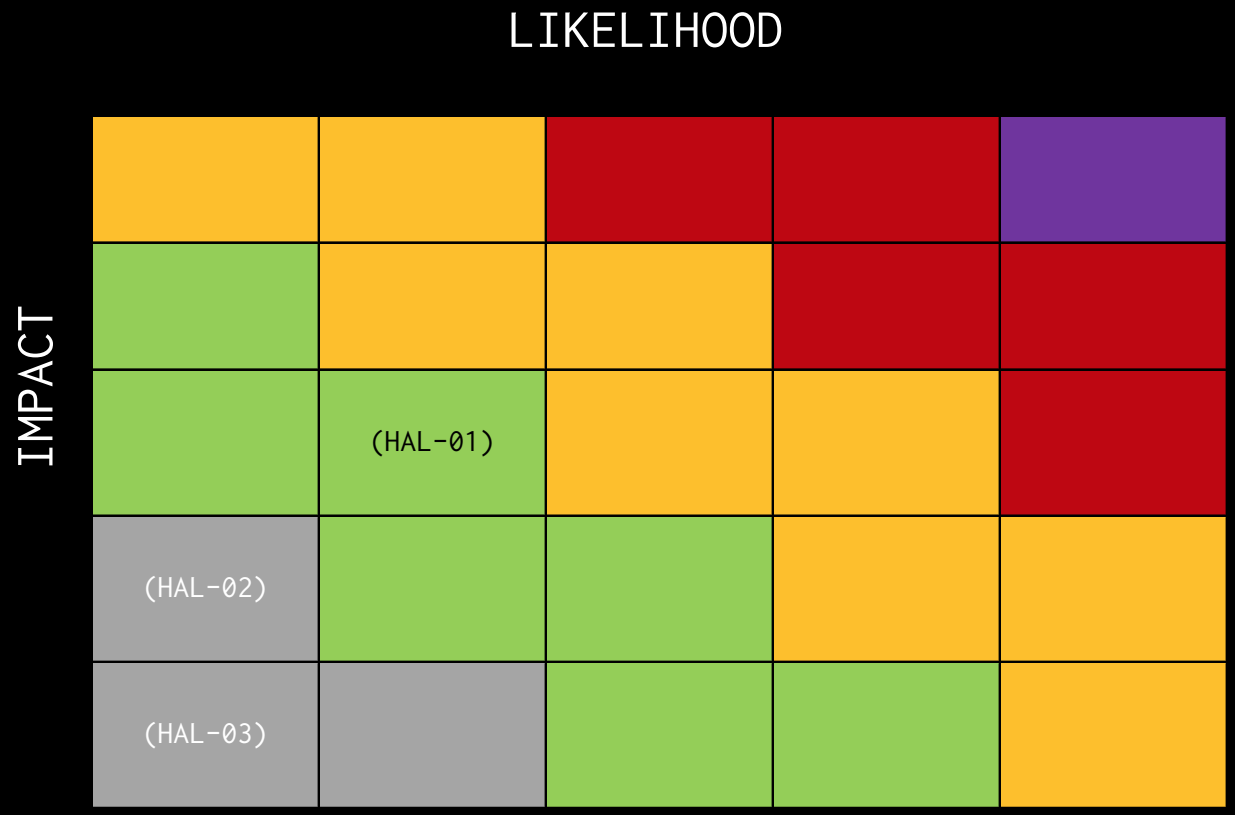
IN-SCOPE:
The security assessment was scoped to the following smart contracts:

- FTMStaking.sol
- SFCPenalty.sol
- sFTMx.sol
- ValidatorPicker.sol
- Vault.sol

Commit ID: ab6fc3ce923a471e920f18a7aa58061b672615ac

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	1	2



EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - OWNER CAN RENOUNCE OWNERSHIP	Low	SOLVED - 04/25/2022
HAL02 - MISSING EVENTS FOR OWNER ONLY FUNCTIONS THAT CHANGE CRITICAL PARAMETERS	Informational	ACKNOWLEDGED
HAL03 - ZERO ADDRESS NOT CHECKED	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) OWNER CAN RENOUNCE OWNERSHIP - LOW

Description:

The Owner of the contract is usually the account that deploys the contract. As a result, the Owner can perform some privileged functions. In the FTMStaking.sol contract, the renounceOwnership function is used to renounce the Owner permission. Renouncing ownership before transferring would result in the contract having no Owner, eliminating the ability to call privileged functions.

Code Location:

Listing 1: FTMStaking.sol (Line 21)

```
21 contract FTMStaking is Initializable, OwnableUpgradeable,
↳ UUPSUpgradeable {
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended that the Owner cannot call renounceOwnership without first transferring Ownership to another address. In addition, if a multi-signature wallet is used, the call to the renounceOwnership function should be confirmed for two or more users.

Remediation Plan:

SOLVED: The issue was solved by using a multi-signature wallet to call `renounceOwnership` function.

3.2 (HAL-02) MISSING EVENTS FOR ADMIN ONLY FUNCTIONS THAT CHANGE CRITICAL PARAMETERS - INFORMATIONAL

Description:

Admin-only functions that change critical parameters should emit events. Events allow you to capture changed parameters so that off-chain tools/interfaces can register those changes.

Code Location:

Listing 2: FTMStaking.sol

```
519     function setTreasury(address newTreasury) external onlyOwner {
520         require(newTreasury != address(0), "ERR_INVALID_VALUE");
521         treasury = newTreasury;
522     }
```

Listing 3: FTMStaking.sol

```
528     function setProtocolFeeBIPS(uint256 newFeeBIPS) external
 ↳ onlyOwner {
529         require(newFeeBIPS <= 10_000, "ERR_INVALID_VALUE");
530         protocolFeeBIPS = newFeeBIPS;
531     }
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendation:

Add events to all admin functions that change critical parameters.

Remediation Plan:

ACKNOWLEDGED: The Staderlabs team acknowledged this issue. However, the functions setTreasury and setProtocolFeeBIPS are not being used by any off-chain process that impacts the protocol and these changes can only be done by using a multi-signature wallet. Therefore, there is no risk to the functioning of the protocol.

3.3 (HAL-03) ZERO ADDRESS NOT CHECKED - INFORMATIONAL

Description:

The updateOwner function within the contract Vault.sol is not verifying that the newOwner parameter is not the zero address to avoid having issues when using the owner-only functions.

Code Location:

Listing 4: Vault.sol (Line 159)

```
158     function updateOwner(address newOwner) external onlyOwner {
159         owner = newOwner;
160     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

When setting an address variable, always ensure the value is not zero.

Remediation Plan:

ACKNOWLEDGED: The Staderlabs team acknowledged this issue.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their ABI and binary formats, Slither was run on the all-scoped contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Slither results:

FTMStaking.sol

```
FTMStaking.claimRewardsAll() (contracts/FTMStaking.sol#668-695) sends eth to arbitrary user
        Dangerous calls:
        - address(treasury).transfer(protocolFee) (contracts/FTMStaking.sol#693)
FTMStaking._lockVault(address,uint256,uint256) (contracts/FTMStaking.sol#764-771) sends eth to arbitrary user
        Dangerous calls:
        - Vault(vault).delegate{value: amount}() (contracts/FTMStaking.sol#769)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

ERC1967UpgradeUpgradeable._functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#198-204) uses delegatecall to a input-controlled function
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall

OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#87) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36)
UUPSUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#187) shadows:
        - ERC1967UpgradeUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

FTMStaking (contracts/FTMStaking.sol#21-976) is an upgradeable contract that does not protect its initiliaze functions: FTMStaking.initialize(IERC20Burnable,ISFC,uint256,uint256,uint256) (contracts/FTMStaking.sol#1
eable.upgradeTo(address) (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#72-75)UUPSUpgradeable.upgradeToAndCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/pro
thub.com/crytic/slither/wiki/Detector-Documentation#unprotected-upgradeable-contract

FTMStaking._undelegate(address,uint256,uint256,uint256) (contracts/FTMStaking.sol#780-858) uses a dangerous strict equality:
        - totalAmountToUndelegate == 0 (contracts/FTMStaking.sol#807)
FTMStaking.getExchangeRate() (contracts/FTMStaking.sol#277-285) uses a dangerous strict equality:
        - totalFTM == 0 || totalFTMx == 0 (contracts/FTMStaking.sol#281)
FTMStaking.getFTMxAmountForFTM(uint256,bool) (contracts/FTMStaking.sol#292-308) uses a dangerous strict equality:
        - totalFTM == 0 || totalFTMx == 0 (contracts/FTMStaking.sol#304)
FTMStaking.pickVaultsToUndelegate(uint256) (contracts/FTMStaking.sol#366-423) uses a dangerous strict equality:
        - amountToUndelegate == amount && amountToUndelegate == amountToReduce && amountToUnlock != amount (contracts/FTMStaking.sol#405-407)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in FTMStaking._undelegate(address,uint256,uint256,uint256) (contracts/FTMStaking.sol#780-858):
        External calls:
        - FTMX.burnFrom(user,amountFTMx) (contracts/FTMStaking.sol#805)
        State variables written after the call(s):
        - request.poolAmount = amount (contracts/FTMStaking.sol#810)
        - request.poolAmount = amount - totalAmountToUndelegate (contracts/FTMStaking.sol#851)
        - request.undelegateAmount = totalAmountToUndelegate (contracts/FTMStaking.sol#852)
        - request.penalty = penalty - implicitPenalty (contracts/FTMStaking.sol#856)
        - _addToPendingWithdrawal(amount) (contracts/FTMStaking.sol#809)
                - ftmPendingWithdrawal += amount (contracts/FTMStaking.sol#879)
        - _addToPendingWithdrawal(amount - totalAmountToUndelegate) (contracts/FTMStaking.sol#813)
                - ftmPendingWithdrawal += amount (contracts/FTMStaking.sol#879)
Reentrancy in FTMStaking._undelegate(address,uint256,uint256,uint256) (contracts/FTMStaking.sol#780-858):
        External calls:
        - FTMX.burnFrom(user,amountFTMx) (contracts/FTMStaking.sol#805)
        - _unlockAndUndelegateVault(info[i].vault,wrID,info[i].amountToUnlock,info[i].amountToUndelegate) (contracts/FTMStaking.sol#826-831)
                - Vault(vault).unlock(amountToUnlock) (contracts/FTMStaking.sol#866)
                - Vault(vault).undelegate(wrID,amountToUndelegate) (contracts/FTMStaking.sol#867)
        State variables written after the call(s):
        - delete _allVaults[vaultPtr] (contracts/FTMStaking.sol#837)
        - request.info.push(UndelegateInfo(info[i].vault,info[i].amountToUnlock,info[i].amountToUndelegate)) (contracts/FTMStaking.sol#842-848)
        - _decrementVaultPtr() (contracts/FTMStaking.sol#839)
                - currentVaultPtr = _decrementWithMod(currentVaultPtr,maxVaultCount) (contracts/FTMStaking.sol#891)
Reentrancy in FTMStaking.harvestVault(uint256) (contracts/FTMStaking.sol#701-722):
        External calls:
        - Vault(vault).undelegate(0,$FC.getStake(vault,toValidatorID)) (contracts/FTMStaking.sol#711)
        - _claim(vault) (contracts/FTMStaking.sol#712)
                - Vault(vault).claimRewards() (contracts/FTMStaking.sol#774-776)
        State variables written after the call(s):
        - delete _allVaults[vaultIndex] (contracts/FTMStaking.sol#718)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Vault.currentStakeValue().penalty (contracts/Vault.sol#51) is a local variable never initialized
FTMStaking.withdraw(uint256,uint256).actualAmountUndelegated (contracts/FTMStaking.sol#628) is a local variable never initialized
FTMStaking.pickVaultsToUndelegate(uint256).emptyCount (contracts/FTMStaking.sol#381) is a local variable never initialized
FTMStaking.calculatePenalty(uint256).totalPenalty (contracts/FTMStaking.sol#335) is a local variable never initialized
FTMStaking.calculatePenalty(uint256).totalStake (contracts/FTMStaking.sol#334) is a local variable never initialized
ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool).slot (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#98) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

```
ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#87-105) ignores return value by IERC1822ProxiableUpg
s/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#98-102)
Vault.unlock(uint256) (contracts/Vault.sol#100-102) ignores return value by SFC.unlockStake(toValidatorID,amount) (contracts/Vault.sol#101)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

SFCPenalty._genResult(ISFC,uint256,uint256,uint256,uint256,uint256,uint256,uint256).fullReward (contracts/libraries/SFCPenalty.sol#234) is written in both
        fullReward = _newRewardsOf(SFC,lockedStake,toValidatorID,stashedUntil,lockedUntil) (contracts/libraries/SFCPenalty.sol#236-242)
        fullReward = _newRewardsOf(SFC,unlockedStake,toValidatorID,stashedUntil,lockedUntil) (contracts/libraries/SFCPenalty.sol#245-251)
SFCPenalty._genResult(ISFC,uint256,uint256,uint256,uint256,uint256,uint256,uint256).fullReward (contracts/libraries/SFCPenalty.sol#234) is written in both
        fullReward = _newRewardsOf(SFC,unlockedStake,toValidatorID,stashedUntil,lockedUntil) (contracts/libraries/SFCPenalty.sol#245-251)
        fullReward = _newRewardsOf(SFC,wholeStake,toValidatorID,lockedUntil,payableUntil) (contracts/libraries/SFCPenalty.sol#254-260)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write

Vault.updateOwner(address) (contracts/Vault.sol#158-160) should emit an event for:
        - owner = newOwner (contracts/Vault.sol#159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

FTMStaking.setProtocolFeeBIPS(uint256) (contracts/FTMStaking.sol#539-542) should emit an event for:
        - protocolFeeBIPS = newFeeBIPS (contracts/FTMStaking.sol#541)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Vault.constructor(ISFC,uint256).auth (contracts/Vault.sol#30) lacks a zero-check on :
                - toValidator = auth (contracts/Vault.sol#32)
Vault.updateOwner(address).newOwner (contracts/Vault.sol#158) lacks a zero-check on :
                - owner = newOwner (contracts/Vault.sol#159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

FTMStaking.totalFTMWorth() (contracts/FTMStaking.sol#255-272) has external calls inside a loop: total += Vault(vault).currentStakeValue() (contracts/FTMStaking.sol#261)
FTMStaking.totalFTMWorth() (contracts/FTMStaking.sol#255-272) has external calls inside a loop: total += Vault(vault_scope_1).currentStakeValue() (contracts/FTMStaking.sol#268)
FTMStaking.calculatePenalty(uint256) (contracts/FTMStaking.sol#317-357) has external calls inside a loop: toValidatorID = Vault(vault).toValidatorID() (contracts/FTMStaking.sol#340)
FTMStaking.calculatePenalty(uint256) (contracts/FTMStaking.sol#317-357) has external calls inside a loop: totalStake += SFC.getStake(vault,toValidatorID) (contracts/FTMStaking.sol#341)
FTMStaking.calculatePenalty(uint256) (contracts/FTMStaking.sol#317-357) has external calls inside a loop: SFC.isLockedUp(vault,toValidatorID) (contracts/FTMStaking.sol#342)
FTMStaking.calculatePenalty(uint256) (contracts/FTMStaking.sol#317-357) has external calls inside a loop: vaultLockedAmount = Vault(vault).getLockedStake() (contracts/FTMStaking.sol#343)
FTMStaking._getAmountsAfterPenalty(address,uint256) (contracts/FTMStaking.sol#938-970) has external calls inside a loop: toValidatorID = Vault(vault).toValidatorID() (contracts/FTMStaking.sol#947)
FTMStaking._getAmountsAfterPenalty(address,uint256) (contracts/FTMStaking.sol#938-970) has external calls inside a loop: vaultLockedAmount = Vault(vault).getLockedStake() (contracts/FTMStaking.sol#948)
SFCPenalty._isLockedUpAtEpoch(ISFC,address,uint256,uint256) (contracts/libraries/SFCPenalty.sol#74-86) has external calls inside a loop: (fromEpoch,endTime) = SFC.getLockupInfo(delegator,toValidatorID) (contracts/l
SFCPenalty._epochEndTime(ISFC,uint256) (contracts/libraries/SFCPenalty.sol#64-72) has external calls inside a loop: (endTime) = SFC.getEpochSnapshot(epoch) (contracts/libraries/SFCPenalty.sol#69)
SFCPenalty._newRewardsOf(ISFC,uint256,uint256,uint256,uint256) (contracts/libraries/SFCPenalty.sol#120-148) has external calls inside a loop: stashedRate = SFC.getEpochAccumulatedRewardPerToken(fromEpoch,toValidato
SFCPenalty._newRewardsOf(ISFC,uint256,uint256,uint256,uint256) (contracts/libraries/SFCPenalty.sol#120-148) has external calls inside a loop: currentRate = SFC.getEpochAccumulatedRewardPerToken(toEpoch,toValidatorI
SFCPenalty._newRewards(ISFC,address,uint256) (contracts/libraries/SFCPenalty.sol#168-221) has external calls inside a loop: stashedUntil = SFC.stashedRewardsUntilEpoch(delegator,toValidatorID) (contracts/libraries/
SFCPenalty._highestPayableEpoch(ISFC,uint256) (contracts/libraries/SFCPenalty.sol#46-62) has external calls inside a loop: (deactivatedEpoch) = SFC.getValidator(validatorID) (contracts/libraries/SFCPenalty.sol#51)
SFCPenalty._highestPayableEpoch(ISFC,uint256) (contracts/libraries/SFCPenalty.sol#46-62) has external calls inside a loop: currentSealedEpoch = SFC.currentSealedEpoch() (contracts/libraries/SFCPenalty.sol#53)
SFCPenalty._highestLockupEpoch(ISFC,address,uint256) (contracts/libraries/SFCPenalty.sol#88-118) has external calls inside a loop: (fromEpoch) = SFC.getLockupInfo(delegator,validatorID) (contracts/libraries/SFCPena
SFCPenalty._highestLockupEpoch(ISFC,address,uint256) (contracts/libraries/SFCPenalty.sol#88-118) has external calls inside a loop: r = SFC.currentSealedEpoch() (contracts/libraries/SFCPenalty.sol#96)
SFCPenalty._newRewards(ISFC,address,uint256) (contracts/libraries/SFCPenalty.sol#168-221) has external calls inside a loop: (lockedStake,duration) = SFC.getLockupInfo(delegator,toValidatorID) (contracts/libraries/S
SFCPenalty._newRewards(ISFC,address,uint256) (contracts/libraries/SFCPenalty.sol#168-221) has external calls inside a loop: wholeStake = SFC.getStake(delegator,toValidatorID) (contracts/libraries/SFCPenalty.sol#203
SFCPenalty.getUnlockPenalty(ISFC,address,uint256,uint256,uint256) (contracts/libraries/SFCPenalty.sol#17-44) has external calls inside a loop: (lockupExtraReward,lockupBaseReward) = SFC.getStashedLockupRewards(vaul
26)
FTMStaking._withdrawVault(address,uint256,bool) (contracts/FTMStaking.sol#870-876) has external calls inside a loop: Vault(vault).withdraw(wrID,withdrawAll) (contracts/FTMStaking.sol#875)
FTMStaking._claim(address) (contracts/FTMStaking.sol#773-778) has external calls inside a loop: Vault(vault).claimRewards() (contracts/FTMStaking.sol#774-776)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Variable 'ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool).slot (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#98)' in ERC1967UpgradeUpgradeable._up
enzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#87-105) potentially used before declaration: require(bool,string)(slot == _IMPLEMENTATION_SLOT,ERC1967Upgrade: unsupported proxiableUUID)
/ERC1967/ERC1967UpgradeUpgradeable.sol#99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in FTMStaking._undelegate(address,uint256,uint256) (contracts/FTMStaking.sol#780-858):
        External calls:
        - FTMX.burnFrom(user,amountFTMx) (contracts/FTMStaking.sol#805)
        - _unlockAndUndelegateVault(info[i].vault,wrID,info[i].amountToUnlock,info[i].amountToUndelegate) (contracts/FTMStaking.sol#826-831)
                - Vault(vault).unlock(amountToUnlock) (contracts/FTMStaking.sol#866)
                - Vault(vault).undelegate(wrID,amountToUndelegate) (contracts/FTMStaking.sol#867)
        State variables written after the call(s):
        - _decrementVaultCount() (contracts/FTMStaking.sol#840)
                - currentVaultCount -= 1 (contracts/FTMStaking.sol#982)
Reentrancy in FTMStaking.harvestVault(uint256) (contracts/FTMStaking.sol#701-722):
        External calls:
        - Vault(vault).undelegate(0,SFC.getStake(vault,toValidatorID)) (contracts/FTMStaking.sol#711)
        - _claim(vault) (contracts/FTMStaking.sol#712)
                - Vault(vault).claimRewards() (contracts/FTMStaking.sol#774-776)
        State variables written after the call(s):
        - _maturedVaults.push(vault) (contracts/FTMStaking.sol#715)
        - _decrementVaultCount() (contracts/FTMStaking.sol#719)
                - currentVaultCount -= 1 (contracts/FTMStaking.sol#982)
Reentrancy in FTMStaking.lock(uint256) (contracts/FTMStaking.sol#434-447):
        External calls:
        - (toValidatorID,lockupDuration) = validatorPicker.getNextValidatorInfo(amount) (contracts/FTMStaking.sol#440-441)
        State variables written after the call(s):
        - newVault = _createVault(toValidatorID) (contracts/FTMStaking.sol#443)
                - _allVaults[currentVaultPtr] = vault (contracts/FTMStaking.sol#758)
        - newVault = _createVault(toValidatorID) (contracts/FTMStaking.sol#443)
                - currentVaultCount += 1 (contracts/FTMStaking.sol#886)
        - newVault = _createVault(toValidatorID) (contracts/FTMStaking.sol#443)
                - currentVaultPtr = _incrementWithMod(currentVaultPtr,maxVaultCount) (contracts/FTMStaking.sol#887)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in FTMStaking.deposit() (contracts/FTMStaking.sol#551-562):
        External calls:
        - FTMX.mint(msg.sender,ftmxAmount) (contracts/FTMStaking.sol#559)
        Event emitted after the call(s):
        - LogDeposited(msg.sender,msg.value,ftmxAmount) (contracts/FTMStaking.sol#561)
Reentrancy in FTMStaking.harvestVault(uint256) (contracts/FTMStaking.sol#701-722):
        External calls:
        - Vault(vault).undelegate(0,SFC.getStake(vault,toValidatorID)) (contracts/FTMStaking.sol#711)
        - _claim(vault) (contracts/FTMStaking.sol#712)
                - Vault(vault).claimRewards() (contracts/FTMStaking.sol#774-776)
        Event emitted after the call(s):
        - LogVaultHarvested(vault,_maturedVaults.length - 1) (contracts/FTMStaking.sol#721)
Reentrancy in FTMStaking.lock(uint256) (contracts/FTMStaking.sol#434-447):
        External calls:
        - (toValidatorID,lockupDuration) = validatorPicker.getNextValidatorInfo(amount) (contracts/FTMStaking.sol#440-441)
        - _lockVault(newVault,lockupDuration,amount) (contracts/FTMStaking.sol#444)
                - Vault(vault).delegate{value: amount}() (contracts/FTMStaking.sol#769)
                - Vault(vault).lockStake(lockupDuration,amount) (contracts/FTMStaking.sol#770)
        External calls sending eth:
        - _lockVault(newVault,lockupDuration,amount) (contracts/FTMStaking.sol#444)
                - Vault(vault).delegate{value: amount}() (contracts/FTMStaking.sol#769)
        Event emitted after the call(s):
        - LogLocked(newVault,lockupDuration,amount) (contracts/FTMStaking.sol#446)
Reentrancy in FTMStaking.undelegate(uint256,uint256,uint256) (contracts/FTMStaking.sol#574-584):
        External calls:
        - _undelegate(msg.sender,wrID,amountFTMx,minAmountFTM) (contracts/FTMStaking.sol#581)
                - Vault(vault).unlock(amountToUnlock) (contracts/FTMStaking.sol#866)
                - Vault(vault).undelegate(wrID,amountToUndelegate) (contracts/FTMStaking.sol#867)
                - FTMX.burnFrom(user,amountFTMx) (contracts/FTMStaking.sol#805)
        Event emitted after the call(s):
        - LogUndelegated(msg.sender,wrID,amountFTMx) (contracts/FTMStaking.sol#583)
Reentrancy in FTMStaking.updateVaultOwner(address,address) (contracts/FTMStaking.sol#481-488):
        External calls:
        - Vault(vault).updateOwner(newOwner) (contracts/FTMStaking.sol#486)
        Event emitted after the call(s):
        - LogVaultOwnerUpdated(msg.sender,vault,newOwner) (contracts/FTMStaking.sol#487)
Reentrancy in FTMStaking.withdrawMatured(uint256) (contracts/FTMStaking.sol#728-740):
        External calls:
        - _withdrawVault(vault,0,true) (contracts/FTMStaking.sol#737)
                - Vault(vault).withdraw(wrID,withdrawAll) (contracts/FTMStaking.sol#875)
        Event emitted after the call(s):
        - LogVaultWithdrawn(vault) (contracts/FTMStaking.sol#739)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

FTMStaking.lock(uint256) (contracts/FTMStaking.sol#434-447) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_now() >= nextEligibleTimestamp,ERR_WAIT_FOR_NEXT_EPOCH) (contracts/FTMStaking.sol#435)
FTMStaking.withdraw(uint256,uint256) (contracts/FTMStaking.sol#605-659) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_now() >= request.requestTime + withdrawalDelay,ERR_NOT_ENOUGH_TIME_PASSED) (contracts/FTMStaking.sol#611-614)
FTMStaking._undelegate(address,uint256,uint256,uint256) (contracts/FTMStaking.sol#780-858) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(request.requestTime == 0,ERR_WRID_ALREADY_USED) (contracts/FTMStaking.sol#798)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) uses assembly
```

```
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#186-189)
StorageSlotUpgradeable.getAddressSlot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#52-56) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#53-55)
StorageSlotUpgradeable.getBooleanSlot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#61-65) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#62-64)
StorageSlotUpgradeable.getBytes32Slot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#70-74) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#71-73)
StorageSlotUpgradeable.getUint256Slot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#79-83) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#80-82)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.2', '^0.8.7']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/interfaces/draft-IERC1822Upgradeable.sol#4)
        - ^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/beacon/IBeaconUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.7 (contracts/FTMStaking.sol#2)
        - ^0.8.7 (contracts/Vault.sol#2)
        - ^0.8.7 (contracts/interfaces/IERC20Burnable.sol#2)
        - ^0.8.7 (contracts/interfaces/ISFC.sol#2)
        - ^0.8.7 (contracts/interfaces/IValidatorPicker.sol#2)
        - ^0.8.7 (contracts/libraries/SFCPenalty.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AddressUpgradeable.functionCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#95-101) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65) is never used and should be removed
ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is never used and should be removed
ContextUpgradeable._msgData() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#27-29) is never used and should be removed
ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init() (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#21-22) is never used and should be removed
ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#24-25) is never used and should be removed
ERC1967UpgradeUpgradeable._changeAdmin(address) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#139-142) is never used and should be removed
ERC1967UpgradeUpgradeable._getAdmin() (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#122-124) is never used and should be removed
ERC1967UpgradeUpgradeable._getBeacon() (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#158-160) is never used and should be removed
ERC1967UpgradeUpgradeable._setAdmin(address) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#129-132) is never used and should be removed
ERC1967UpgradeUpgradeable._setBeacon(address) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#165-172) is never used and should be removed
ERC1967UpgradeUpgradeable._upgradeBeaconToAndCall(address,bytes,bool) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#180-190) is never used and should be removed
StorageSlotUpgradeable.getBytes32Slot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#70-74) is never used and should be removed
StorageSlotUpgradeable.getUint256Slot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#79-83) is never used and should be removed
UUPSUpgradeable.__UUPSUpgradeable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#26-27) is never used and should be removed
UUPSUpgradeable._authorizeUpgrade(address) (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#100) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/interfaces/draft-IERC1822Upgradeable.sol#4) allows old versions
Pragma version^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/beacon/IBeaconUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in ERC1967UpgradeUpgradeable._functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#198-204):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#202)
Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function OwnableUpgradeable.__Ownable_init() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#29-31) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#33-35) is not in mixedCase
Variable OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#87) is not in mixedCase
Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init() (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#21-22) is not in mixedCase
Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#24-25) is not in mixedCase
Variable ERC1967UpgradeUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#211) is not in mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init() (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#23-24) is not in mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#26-27) is not in mixedCase
Variable UUPSUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#107) is not in mixedCase
Variable UUPSUpgradeable.__self (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#29) is not in mixedCase
Function ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36) is not in mixedCase
Parameter FTMStaking.initialize(IERC20Burnable,ISFC,uint256,uint256,uint256)._ftmx_ (contracts/FTMStaking.sol#177) is not in mixedCase
Parameter FTMStaking.initialize(IERC20Burnable,ISFC,uint256,uint256,uint256)._sfc_ (contracts/FTMStaking.sol#178) is not in mixedCase
Variable FTMStaking.FTMX (contracts/FTMStaking.sol#46) is not in mixedCase
Variable FTMStaking.SFC (contracts/FTMStaking.sol#51) is not in mixedCase
Variable Vault.SFC (contracts/Vault.sol#15) is not in mixedCase
Parameter SFCPenalty.getUnlockPenalty(ISFC,address,uint256,uint256,uint256).SFC (contracts/libraries/SFCPenalty.sol#18) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in FTMStaking.withdraw(uint256,uint256) (contracts/FTMStaking.sol#605-659):
        External calls:
        - address(user).transfer(totalAmount) (contracts/FTMStaking.sol#656)
        Event emitted after the call(s):
        - LogWithdrawn(user,wrID,totalAmount,bitmaskToSkip) (contracts/FTMStaking.sol#658)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

UUPSUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#107) is never used in FTMStaking (contracts/FTMStaking.sol#21-976)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#59-61)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#67-70)
initialize(IERC20Burnable,ISFC,uint256,uint256,uint256) should be declared external:
        - FTMStaking.initialize(IERC20Burnable,ISFC,uint256,uint256,uint256) (contracts/FTMStaking.sol#176-196)
getUnlockPenalty(ISFC,address,uint256,uint256,uint256) should be declared external:
        - SFCPenalty.getUnlockPenalty(ISFC,address,uint256,uint256,uint256) (contracts/libraries/SFCPenalty.sol#17-44)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

SFCPenalty.sol

```
SFCPenalty._genResult(ISFC,uint256,uint256,uint256,uint256,uint256,uint256).fullReward (contracts/libraries/SFCPenalty.sol#234) is written in both
        fullReward = _newRewardsOf(SFC,lockedStake,toValidatorID,stashedUntil,lockedUntil) (contracts/libraries/SFCPenalty.sol#236-242)
        fullReward = _newRewardsOf(SFC,unlockedStake,toValidatorID,stashedUntil,lockedUntil) (contracts/libraries/SFCPenalty.sol#245-251)
SFCPenalty._genResult(ISFC,uint256,uint256,uint256,uint256,uint256,uint256).fullReward (contracts/libraries/SFCPenalty.sol#234) is written in both
        fullReward = _newRewardsOf(SFC,unlockedStake,toValidatorID,stashedUntil,lockedUntil) (contracts/libraries/SFCPenalty.sol#245-251)
        fullReward = _newRewardsOf(SFC,wholeStake,toValidatorID,lockedUntil,payableUntil) (contracts/libraries/SFCPenalty.sol#254-260)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write

solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter SFCPenalty.getUnlockPenalty(ISFC,address,uint256,uint256,uint256).SFC (contracts/libraries/SFCPenalty.sol#18) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

getUnlockPenalty(ISFC,address,uint256,uint256,uint256) should be declared external:
        - SFCPenalty.getUnlockPenalty(ISFC,address,uint256,uint256,uint256) (contracts/libraries/SFCPenalty.sol#17-44)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

sFTMx.sol

```
AccessControl._setRoleAdmin(bytes32,bytes32) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#194-198) is never used and should be removed
AccessControl._setupRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#185-187) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#40-51) is never used and should be removed
Strings.toString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#15-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Contract sFTMx (contracts/sFTMx.sol#13-42) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#130-132)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#143-145)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#161-165)
name() should be declared external:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
totalSupply() should be declared external:
        - ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-117)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#136-140)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#201-210)
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#20-22)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#35-38)
        - sFTMx.burnFrom(address,uint256) (contracts/sFTMx.sol#35-41)
mint(address,uint256) should be declared external:
        - sFTMx.mint(address,uint256) (contracts/sFTMx.sol#26-28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

ValidatorPicker.sol

```
ValidatorPicker.setNextValidatorInfo(uint256,uint256) (contracts/ValidatorPicker.sol#35-41) should emit an event for:
        - _toValidatorID = toValidatorID (contracts/ValidatorPicker.sol#39)
        - _lockupDuration = lockupDuration (contracts/ValidatorPicker.sol#40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.7']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.7 (contracts/ValidatorPicker.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Vault.sol

```
Vault.currentStakeValue().penalty (contracts/Vault.sol#51) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Vault.unlock(uint256) (contracts/Vault.sol#100-102) ignores return value by SFC.unlockStake(toValidatorID,amount) (contracts/Vault.sol#101)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Vault.updateOwner(address) (contracts/Vault.sol#158-160) should emit an event for:
        - owner = newOwner (contracts/Vault.sol#159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

Vault.constructor(ISFC,uint256).auth (contracts/Vault.sol#30) lacks a zero-check on :
                - toValidator = auth (contracts/Vault.sol#32)
Vault.updateOwner(address).newOwner (contracts/Vault.sol#158) lacks a zero-check on :
                - owner = newOwner (contracts/Vault.sol#159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable Vault.SFC (contracts/Vault.sol#15) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

- As a result of the tests carried out with the Slither tool, some results were obtained and reviewed by Halborn. Based on the results reviewed, some vulnerabilities were determined to be false positives. The actual vulnerabilities found by Slither are already included in the report findings.

4.2 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 all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

FTMStaking.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
88	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
90	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

SFCPenalty.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

sFTMx.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

ValidatorPicker.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Vault.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
54	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
56	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
56	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
130	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
172	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
172	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
172	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
172	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered

- The floating pragma flagged by MythX is a false positive, as the pragma is set in the `hardhat.config.js` file to the `0.8.7` version.

THANK YOU FOR CHOOSING

// HALBORN