Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Customer:	FAST Yield
Website:	https://fastyield.app
Platform:	Fantom Blockchain
Language:	Solidity
Date:	October 15th, 2021

Table of contents

Email: audit@EtherAuthority.io

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the FAST Yield Protocol team to perform the Security audit of the FAST Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on October 15th, 2021.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

Audit scope

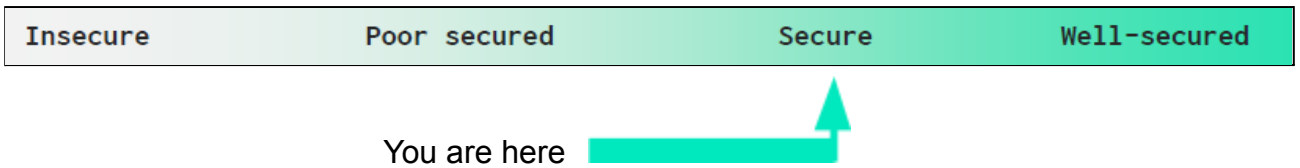
Name	Code Review and Security Analysis Report for FAST Yield Protocol Smart Contracts
Platform	Fantom / Solidity
File 1	FAST.sol
File 1 MD5 Hash	61F3C955FD33693DEC5AD3DA2CE82D4A
File 2	NativeFarm.sol
File 2 MD5 Hash	06F6863014F3FEAA85EDA13AE0DDFCBE
File 3	StrategyV3_Jetswap.sol
File 3 MD5 Hash	5AFA1B9E204435F9FF2075A7A193ED1A
Audit Date	October 15th, 2021

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1: FAST.sol • Decimals: 18 • Name: Fast Yield Token • Symbol: FAST • Initial Supply: 100k • Token minting can be done by the owner (which should be the masterChef) • Token Max Supply: 793,792	**YES, This is valid.**
File 2: NativeFarm.sol • Mints FAST tokens • FAST reward per second: 0.02	**YES, This is valid.**
File 3: StrategyV3_Jetswap.sol • Buy Back Rate: 2% • Buy Back Rate Max: 10% • Buy Back Rate UL: 8% • Controller Fee: 2% • Controller Fee Max: 10% • Controller Fee UL: 3%	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“Secured”**. This token contract does contain owner control, which does not make it fully decentralized.



We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 4 low and some very low level issues. These issues are fixed/acknowledged.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Moderated
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	Passed
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Moderated
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 3 smart contract files. Smart contracts also contain Libraries, Smart contracts inherits and Interfaces. These are compact and well written contracts.

The libraries are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the FAST Protocol.

The FAST Protocol team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are **Not well** commented on smart contracts.

Documentation

We were given a FAST smart contract code in the form of a FTMScan web Link. The hashes of that code are mentioned above in the table.

As mentioned above, code parts are **Not well** commented. So it is not easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects. And their core code blocks are written well.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

FAST.sol

(1) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	mint	external	Missing required error message	Refer audit finding section
3	addMinter	external	Function input parameters lack of check	Refer audit finding section
4	removeMinter	external	Function input parameters lack of check	Refer audit finding section
5	owner	read	Functions should be declared as external	Refer audit finding section
6	onlyOwner	modifier	Passed	No Issue
7	renounceOwnership	write	Functions should be declared as external	Refer audit finding section
8	transferOwnership	write	Functions should be declared as external	Refer audit finding section
9	name	read	Passed	No Issue
10	symbol	read	Passed	No Issue
11	decimals	read	Passed	No Issue
12	totalSupply	read	Passed	No Issue
13	balanceOf	read	Passed	No Issue
14	transfer	write	Passed	No Issue
15	allowance	read	Passed	No Issue
16	approve	write	Passed	No Issue
17	transferFrom	write	Passed	No Issue
18	increaseAllowance	write	Passed	No Issue
19	decreaseAllowance	write	Passed	No Issue
20	_transfer	internal	Passed	No Issue
21	_mint	internal	Passed	No Issue
22	_burn	internal	Passed	No Issue
23	_approve	internal	Passed	No Issue
24	_setupDecimals	internal	Passed	No Issue
25	_beforeTokenTransfer	internal	Passed	No Issue

NativeFarm.sol

(1) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	poolLength	external	Out of Gas	Refer audit finding section
3	add	write	Critical operation lacks event log	Refer audit finding section
4	set	write	Critical operation lacks event log	Refer audit finding section
5	getMultiplier	read	Passed	No Issue
6	pendingNATIVE	external	Passed	No Issue
7	stakedWantTokens	external	Passed	No Issue
8	updatePool	write	Critical operation lacks event log	Refer audit finding section
9	deposit	external	Passed	No Issue
10	withdraw	write	Passed	No Issue
11	withdrawAll	external	Passed	No Issue
12	emergencyWithdraw	external	Passed	No Issue
13	safeNATIVETransfer	internal	Passed	No Issue
14	inCaseTokensGetStuck	external	access only Owner	No Issue
15	owner	read	Passed	No Issue
16	onlyOwner	modifier	Passed	No Issue
17	renounceOwnership	write	access only Owner	No Issue
18	transferOwnership	write	access only Owner	No Issue
19	nonReentrant	modifier	Passed	No Issue

StrategyV3_Jetswap.sol

(1) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	_farm	internal	Passed	No Issue
3	_unfarm	internal	Passed	No Issue
4	onlyAllowGov	modifier	Passed	No Issue
5	deposit	write	Unused variable	Refer audit finding section
6	farm	write	access nonReentrant	No Issue
7	_farm	internal	Passed	No Issue
8	_unfarm	internal	Passed	No Issue
9	withdraw	write	Unused variable	Refer audit finding section
10	_harvest	internal	Passed	No Issue
11	earn	write	Passed	No Issue
12	buyBack	internal	Passed	No Issue
13	distributeFees	internal	Passed	No Issue
14	convertDustToEarned	write	Passed	No Issue
15	pause	write	access only AllowGov	No Issue
16	unpause	external	access only AllowGov	No Issue
17	setEntranceFeeFactor	write	access only AllowGov	No Issue
18	setControllerFee	write	access only AllowGov	No Issue
19	setGov	write	access only AllowGov	No Issue
20	setDepositFeeFactor	write	access only AllowGov	No Issue
21	setWithdrawFeeFactor	write	access only AllowGov	No Issue
22	setbuyBackRate	write	access only AllowGov	No Issue
23	setBuybackRouterAddress	write	access only AllowGov	No Issue
24	inCaseTokensGetStuck	write	access only AllowGov	No Issue
25	_wrapFTM	internal	access only AllowGov	No Issue
26	_safeSwap	internal	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: audit@EtherAuthority.io

Audit Findings

FAST.sol

Critical

No Critical severity vulnerabilities were found.

High

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Function input parameters lack of check:

```
function addMinter(address _minter) external onlyOwner {
    minters[_minter] = true;
}
```

The function addMinter has an input parameter _minter, which should be validated as being a non-zero address (address(0)) to prevent any unintended outcomes due to human error. There are several other places where input validations should be added.

Resolution: We advise adding one “require” condition as below:

require(_minter != address(0), “Invalid address”);

Very Low / Informational / Best practices:

(1) Use the latest solidity version:

```
pragma solidity 0.6.12;
```

Using the latest solidity will prevent any compiler-level bugs.

Resolution: Please use 0.8.9 which is the latest version.

(2) Functions should be declared as external:

Any functions which are not called internally, should be declared as external. It is good practice and it saves some gas as well. We suggest changing the following functions visibility as external:

- ERC20.name()
- ERC20.symbol()
- ERC20.totalSupply()
- ERC20.balanceOf()
- ERC20.transfer()
- ERC20.allowance()
- ERC20.approve()
- ERC20.transferFrom()
- ERC20.increaseAllowance()
- ERC20.decreaseAllowance()
- Ownable.owner()
- Ownable.renounceOwnership()
- Ownable.transferOwnership()

(3) Critical operation lacks event log:

Any significant state changing function should fire an event to properly coordinate with the client. Please consider adding events in the following functions:

- addMinter() - onlyOwner
- removeMinter() - onlyOwner

(4) Ambiguous error message:

```
function mint(address _to, uint256 _amount) external {
    require(minters[msg.sender], "!minter");
    _mint(_to, _amount);
}
```

Here, the message does not clearly state the error.

Resolution: We suggest setting a proper error message to identify failed execution in the function. And this will also increase the readability.

NativeFarm.sol

Critical

No Critical severity vulnerabilities were found.

High

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Critical operation lacks event log:

There are several places in the smart contracts where an event log is missing.

Resolution: Functions are:

- add() - onlyOwner
- set() - onlyOwner
- updatePool()

(2) Function input parameters lack of check:

```
// Add a new lp to the pool. Can only be called by the owner.
// XXX DO NOT add the same LP token more than once. Rewards will be messed up if you do. (Only if want tokens are stored here.)
function add(
    uint256 _allocPoint,
    IERC20 _want,
    address _strat
) external onlyOwner {
    uint256 lastRewardTime = block.timestamp > startTime ? block.timestamp : startTime;
    totalAllocPoint = totalAllocPoint.add(_allocPoint);
    poolInfo.push(
        PoolInfo({
            want: _want,
            allocPoint: _allocPoint,
            lastRewardTime: lastRewardTime,
            accNATIVEPerShare: 0,
            strat: _strat
        })
    );
}
```

As mentioned in comment, code should not allow adding the same LP token more than once, but there is no validation for the "_want" token parameter.

Resolution: we suggest adding a proper input param validations to prevent any discrepancy due to human errors.

Very Low / Informational / Best practices:

(1) Use the latest solidity version:

```
pragma solidity 0.6.12;
```

Using the latest solidity will prevent any compiler-level bugs.

Resolution: Please use 0.8.9 which is the latest version.

(2) Out of Gas :

Pool (PoolInfo) storage has no set length limit.

Same as User (UserInfo) storage has no set length or limit.

Resolution: Set the length of limit for saving GAs, If pool storage and user storage have many blocks it will consume high Gas..

StrategyV3_Jetswap.sol

Critical

No Critical severity vulnerabilities were found.

High

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Unused variable:

```
function deposit(address _userAddress, uint256 _wantAmt)
    public
    virtual
    onlyOwner
    nonReentrant
    whenNotPaused
    returns (uint256)
{
```

```
function withdraw(address _userAddress, uint256 _wantAmt)
    public
    virtual
    onlyOwner
    nonReentrant
    returns (uint256)
{
```

There are 2 functions, which have 2 parameters but are not used anywhere in function code.

Resolution: Functions: withdraw() and deposit()

Remove unused variables from functions..

Very Low / Informational / Best practices:

(1) Use the latest solidity version:

```
pragma solidity 0.6.12;
```

Using the latest solidity will prevent any compiler-level bugs.

Resolution: Please use 0.8.9 which is the latest version.

Centralization

These smart contracts have some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- addMinter: The Fast Owner can add minter.
- removeMinter: The Fast Owner can remove the minter.
- add: The NativeFarm Owner can add a new lp to the pool. Can only be called by the owner.
- set: The NativeFarm Owner can Update the given pool's NATIVE allocation point.
- inCaseTokensGetStuck: The NativeFarm Owner can check !safe, stuck or not.
- deposit: The StrategyV3_Jetswap owner can deposit a fee.
- withdraw: The StrategyV3_Jetswap owner can withdraw a fee.
- pause: The StrategyV3_Jetswap owner can pause.
- unpause: The StrategyV3_Jetswap owner can unpause.
- setEntranceFeeFactor: The StrategyV3_Jetswap owner can set the entrance fee factor.
- setControllerFee: The StrategyV3_Jetswap owner can set controller fee.
- setGov: The StrategyV3_Jetswap owner can set Gov.
- setDepositFeeFactor: The StrategyV3_Jetswap owner can set deposit fee factor.
- setWithdrawFeeFactor: The StrategyV3_Jetswap owner can set withdrawal fee factor.
- setbuyBackRate: The StrategyV3_Jetswap owner can set buy back rate.
- setBuybackRouterAddress: The StrategyV3_Jetswap owner can set buy back router addresses.
- inCaseTokensGetStuck: The StrategyV3_Jetswap owner can get stuck.
- _wrapFTM: The StrategyV3_Jetswap owner can wrapFTM.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts and those issues are not critical ones. So, **it's good to go to production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured".**

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

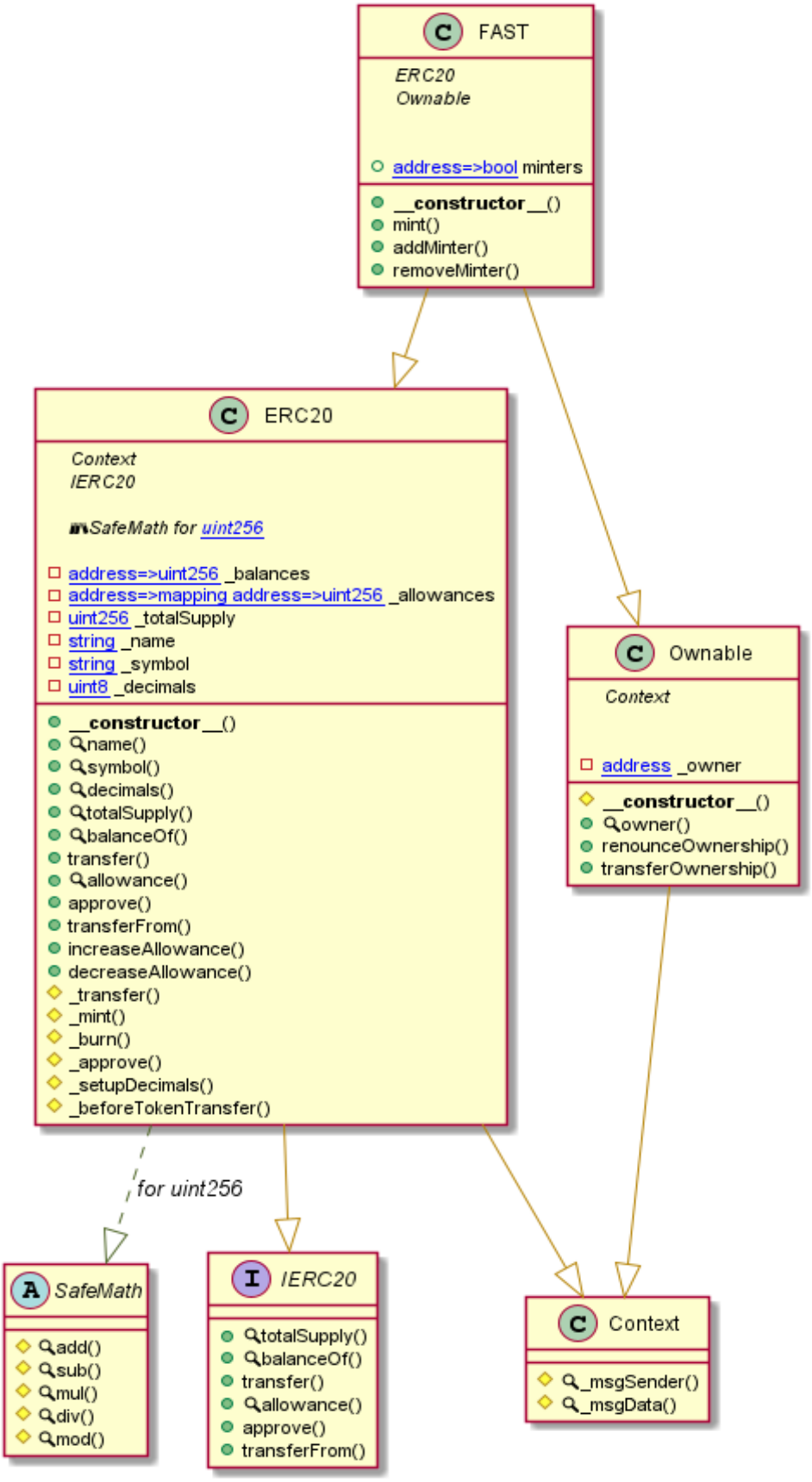
Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

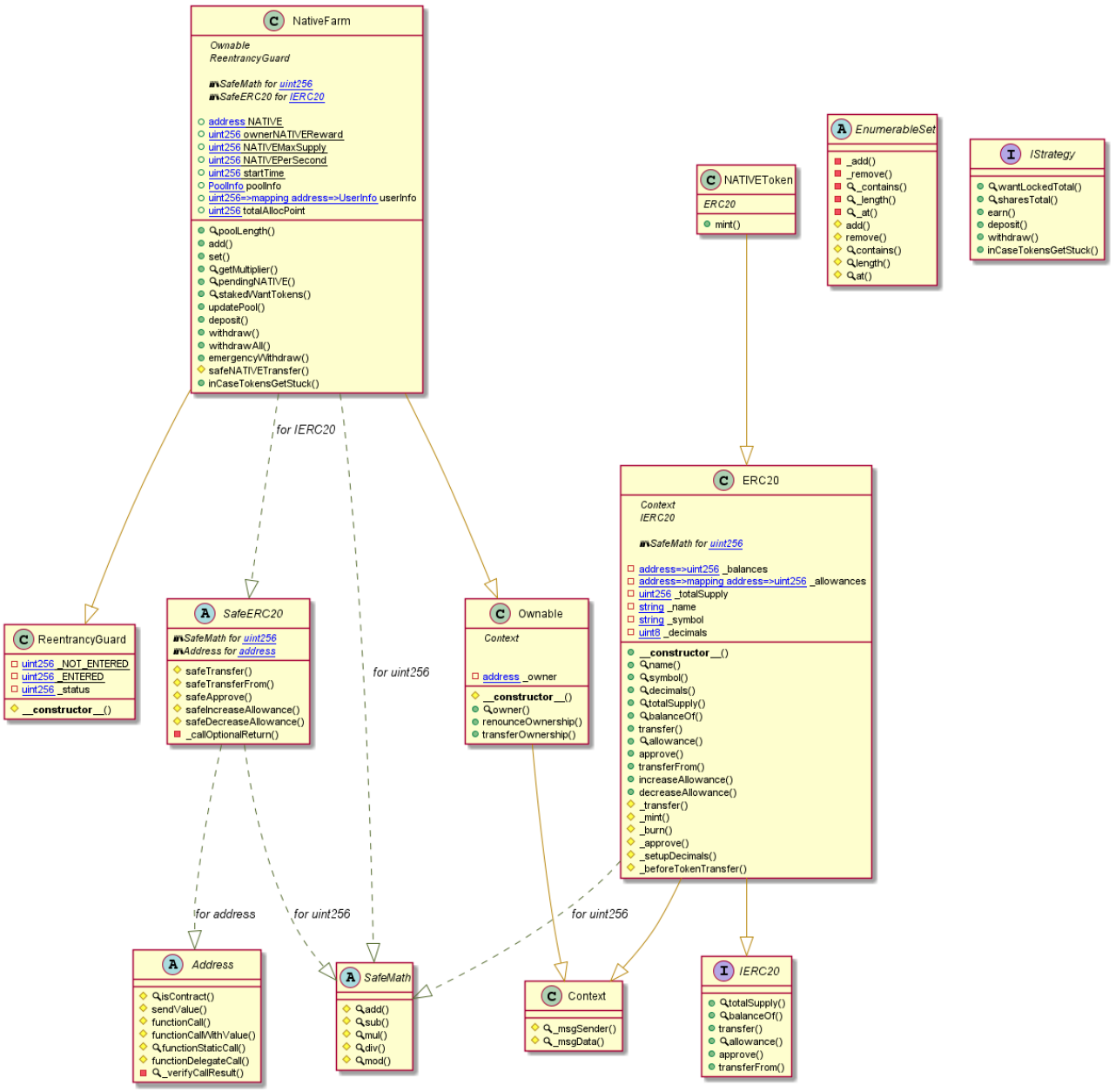
Appendix

Code Flow Diagram - FAST Protocol

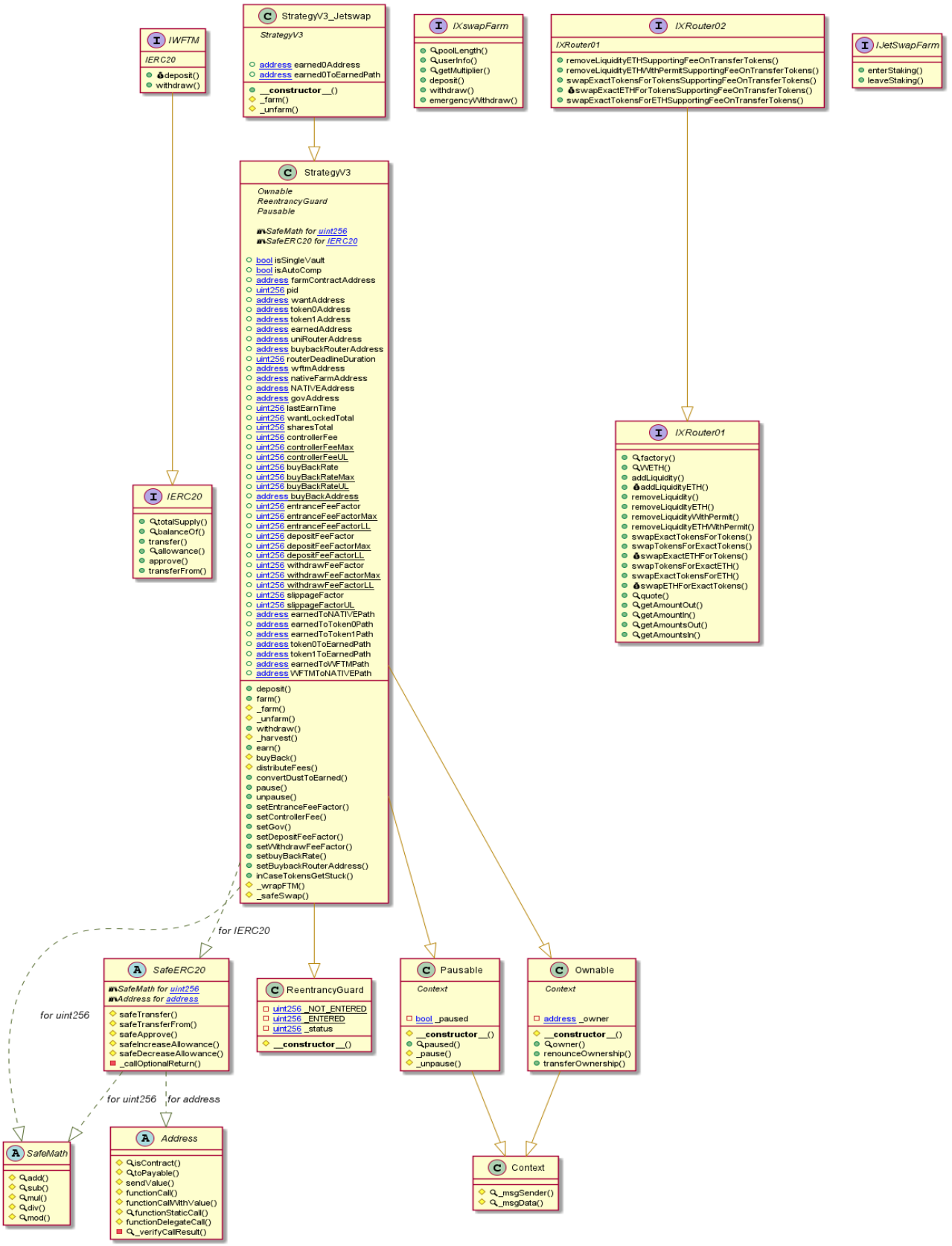
FAST Diagram



NativeFarm Diagram



StrategyV3_Jetswap Diagram



Slither Results Log

Slither log >> FAST.sol

```
INFO:Detectors:
FAST.constructor(string,string,address)._name (FAST.sol#657) shadows:
        - ERC20._name (FAST.sol#268) (state variable)
FAST.constructor(string,string,address)._symbol (FAST.sol#657) shadows:
        - ERC20._symbol (FAST.sol#269) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Context._msgData() (FAST.sol#14-17) is never used and should be removed
ERC20._burn(address,uint256) (FAST.sol#525-536) is never used and should be removed
ERC20._setupDecimals(uint8) (FAST.sol#570-572) is never used and should be removed
SafeMath.div(uint256,uint256) (FAST.sol#109-111) is never used and should be removed
SafeMath.div(uint256,uint256,string) (FAST.sol#125-135) is never used and should be removed
SafeMath.mod(uint256,uint256) (FAST.sol#149-151) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (FAST.sol#165-172) is never used and should be removed
SafeMath.mul(uint256,uint256) (FAST.sol#83-95) is never used and should be removed
SafeMath.sub(uint256,uint256) (FAST.sol#48-50) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Parameter FAST.mint(address,uint256)._to (FAST.sol#662) is not in mixedCase
Parameter FAST.mint(address,uint256)._amount (FAST.sol#662) is not in mixedCase
Parameter FAST.addMinter(address)._minter (FAST.sol#667) is not in mixedCase
Parameter FAST.removeMinter(address)._minter (FAST.sol#671) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (FAST.sol#15)" inContext (FAST.sol#9-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
FAST.constructor(string,string,address) (FAST.sol#657-660) uses literals with too many digits:
        - _mint(msg.sender,10_420_000000_000000_000000) (FAST.sol#658)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (FAST.sol#290-292)
symbol() should be declared external:
        - ERC20.symbol() (FAST.sol#298-300)
decimals() should be declared external:
        - ERC20.decimals() (FAST.sol#315-317)
totalSupply() should be declared external:
```

```
totalSupply() should be declared external:
        - ERC20.totalSupply() (FAST.sol#322-324)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (FAST.sol#329-331)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (FAST.sol#341-349)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (FAST.sol#354-362)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (FAST.sol#371-379)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (FAST.sol#394-409)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (FAST.sol#423-434)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (FAST.sol#450-464)
owner() should be declared external:
        - Ownable.owner() (FAST.sol#615-617)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (FAST.sol#634-637)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (FAST.sol#643-650)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:FAST.sol analyzed (6 contracts with 75 detectors), 31 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> NativeFarm.sol

```
INFO:Detectors:
NativeFarm.safeNATIVETransfer(address,uint256) (NativeFarm.sol#1689-1696) ignores return value by IERC20(NATIVE).transfer(_to,NATIVEBal)
(NativeFarm.sol#1692)
NativeFarm.safeNATIVETransfer(address,uint256) (NativeFarm.sol#1689-1696) ignores return value by IERC20(NATIVE).transfer(_to,_NATIVEAmt)
 (NativeFarm.sol#1694)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
NativeFarm.pendingNATIVE(uint256,address) (NativeFarm.sol#1515-1535) performs a multiplication on the result of a division:
        -NATIVEReward = multiplier.mul(NATIVEPerSecond).mul(pool.allocPoint).div(totalAllocPoint) (NativeFarm.sol#1526-1529)
        -accNATIVEPerShare = accNATIVEPerShare.add(NATIVEReward.mul(1e12).div(sharesTotal)) (NativeFarm.sol#1530-1532)
NativeFarm.updatePool(uint256) (NativeFarm.sol#1556-1585) performs a multiplication on the result of a division:
        -NATIVEReward = multiplier.mul(NATIVEPerSecond).mul(pool.allocPoint).div(totalAllocPoint) (NativeFarm.sol#1570-1573)
        -NATIVEToken(NATIVE).mint(owner(),NATIVEReward.mul(ownerNATIVEReward).div(10000)) (NativeFarm.sol#1575-1578)
NativeFarm.updatePool(uint256) (NativeFarm.sol#1556-1585) performs a multiplication on the result of a division:
        -NATIVEReward = multiplier.mul(NATIVEPerSecond).mul(pool.allocPoint).div(totalAllocPoint) (NativeFarm.sol#1570-1573)
        -pool.accNATIVEPerShare = pool.accNATIVEPerShare.add(NATIVEReward.mul(1e12).div(sharesTotal)) (NativeFarm.sol#1581-1583)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in NativeFarm.deposit(uint256,uint256) (NativeFarm.sol#1588-1616):
        External calls:
        - updatePool(_pid) (NativeFarm.sol#1589)
                - NATIVEToken(NATIVE).mint(owner(),NATIVEReward.mul(ownerNATIVEReward).div(10000)) (NativeFarm.sol#1575-1578)
                - NATIVEToken(NATIVE).mint(address(this),NATIVEReward) (NativeFarm.sol#1579)
        - safeNATIVETransfer(msg.sender,pending) (NativeFarm.sol#1599)
                - IERC20(NATIVE).transfer(_to,NATIVEBal) (NativeFarm.sol#1692)
                - IERC20(NATIVE).transfer(_to,_NATIVEAmt) (NativeFarm.sol#1694)
        - pool.want.safeTransferFrom(address(msg.sender),address(this),_wantAmt) (NativeFarm.sol#1603-1607)
        - pool.want.safeIncreaseAllowance(pool.strat,_wantAmt) (NativeFarm.sol#1609)
        - sharesAdded = IStrategy(poolInfo[_pid].strat).deposit(msg.sender,_wantAmt) (NativeFarm.sol#1610-1611)
        State variables written after the call(s):
        - user.shares = user.shares.add(sharesAdded) (NativeFarm.sol#1612)
        - user.rewardDebt = user.shares.mul(pool.accNATIVEPerShare).div(1e12) (NativeFarm.sol#1614)
Reentrancy in NativeFarm.emergencyWithdraw(uint256) (NativeFarm.sol#1671-1686):
        External calls:
        - IStrategy(poolInfo[_pid].strat).withdraw(msg.sender,amount) (NativeFarm.sol#1680)
        - pool.want.safeTransfer(address(msg.sender),amount) (NativeFarm.sol#1682)
        State variables written after the call(s):
```

```
        State variables written after the call(s):
        - user.shares = 0 (NativeFarm.sol#1684)
        - user.rewardDebt = 0 (NativeFarm.sol#1685)
Reentrancy in NativeFarm.updatePool(uint256) (NativeFarm.sol#1556-1585):
        External calls:
        - NATIVEToken(NATIVE).mint(owner(),NATIVEReward.mul(ownerNATIVEReward).div(10000)) (NativeFarm.sol#1575-1578)
        - NATIVEToken(NATIVE).mint(address(this),NATIVEReward) (NativeFarm.sol#1579)
        State variables written after the call(s):
        - pool.accNATIVEPerShare = pool.accNATIVEPerShare.add(NATIVEReward.mul(1e12).div(sharesTotal)) (NativeFarm.sol#1581-1583)
        - pool.lastRewardTime = block.timestamp (NativeFarm.sol#1584)
Reentrancy in NativeFarm.withdraw(uint256,uint256) (NativeFarm.sol#1619-1664):
        External calls:
        - updatePool(_pid) (NativeFarm.sol#1620)
                - NATIVEToken(NATIVE).mint(owner(),NATIVEReward.mul(ownerNATIVEReward).div(10000)) (NativeFarm.sol#1575-1578)
                - NATIVEToken(NATIVE).mint(address(this),NATIVEReward) (NativeFarm.sol#1579)
        - safeNATIVETransfer(msg.sender,pending) (NativeFarm.sol#1638)
                - IERC20(NATIVE).transfer(_to,NATIVEBal) (NativeFarm.sol#1692)
                - IERC20(NATIVE).transfer(_to,_NATIVEAmt) (NativeFarm.sol#1694)
        - sharesRemoved = IStrategy(poolInfo[_pid].strat).withdraw(msg.sender,_wantAmt) (NativeFarm.sol#1647-1648)
        State variables written after the call(s):
        - user.shares = 0 (NativeFarm.sol#1651)
        - user.shares = user.shares.sub(sharesRemoved) (NativeFarm.sol#1653)
Reentrancy in NativeFarm.withdraw(uint256,uint256) (NativeFarm.sol#1619-1664):
        External calls:
        - updatePool(_pid) (NativeFarm.sol#1620)
                - NATIVEToken(NATIVE).mint(owner(),NATIVEReward.mul(ownerNATIVEReward).div(10000)) (NativeFarm.sol#1575-1578)
                - NATIVEToken(NATIVE).mint(address(this),NATIVEReward) (NativeFarm.sol#1579)
        - safeNATIVETransfer(msg.sender,pending) (NativeFarm.sol#1638)
                - IERC20(NATIVE).transfer(_to,NATIVEBal) (NativeFarm.sol#1692)
                - IERC20(NATIVE).transfer(_to,_NATIVEAmt) (NativeFarm.sol#1694)
        - sharesRemoved = IStrategy(poolInfo[_pid].strat).withdraw(msg.sender,_wantAmt) (NativeFarm.sol#1647-1648)
        - pool.want.safeTransfer(address(msg.sender),_wantAmt) (NativeFarm.sol#1660)
        State variables written after the call(s):
        - user.rewardDebt = user.shares.mul(pool.accNATIVEPerShare).div(1e12) (NativeFarm.sol#1662)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
NativeFarm.emergencyWithdraw(uint256) (NativeFarm.sol#1671-1686) ignores return value by IStrategy(poolInfo[_pid].strat).withdraw(msg.sen
der,amount) (NativeFarm.sol#1680)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in NativeFarm.deposit(uint256,uint256) (NativeFarm.sol#1588-1616):
        External calls:
        - updatePool(_pid) (NativeFarm.sol#1589)
                - NATIVEToken(NATIVE).mint(owner(),NATIVEReward.mul(ownerNATIVEReward).div(10000)) (NativeFarm.sol#1575-1578)
                - NATIVEToken(NATIVE).mint(address(this),NATIVEReward) (NativeFarm.sol#1579)
        - safeNATIVETransfer(msg.sender,pending) (NativeFarm.sol#1599)
                - IERC20(NATIVE).transfer(_to,NATIVEBal) (NativeFarm.sol#1692)
                - IERC20(NATIVE).transfer(_to,_NATIVEAmt) (NativeFarm.sol#1694)
        - pool.want.safeTransferFrom(address(msg.sender),address(this),_wantAmt) (NativeFarm.sol#1603-1607)
        - pool.want.safeIncreaseAllowance(pool.strat,_wantAmt) (NativeFarm.sol#1609)
        - sharesAdded = IStrategy(poolInfo[_pid].strat).deposit(msg.sender,_wantAmt) (NativeFarm.sol#1610-1611)
        Event emitted after the call(s):
        - Deposit(msg.sender,_pid,_wantAmt) (NativeFarm.sol#1615)
Reentrancy in NativeFarm.emergencyWithdraw(uint256) (NativeFarm.sol#1671-1686):
        External calls:
        - IStrategy(poolInfo[_pid].strat).withdraw(msg.sender,amount) (NativeFarm.sol#1680)
        - pool.want.safeTransfer(address(msg.sender),amount) (NativeFarm.sol#1682)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,amount) (NativeFarm.sol#1683)
Reentrancy in NativeFarm.withdraw(uint256,uint256) (NativeFarm.sol#1619-1664):
        External calls:
        - updatePool(_pid) (NativeFarm.sol#1620)
                - NATIVEToken(NATIVE).mint(owner(),NATIVEReward.mul(ownerNATIVEReward).div(10000)) (NativeFarm.sol#1575-1578)
                - NATIVEToken(NATIVE).mint(address(this),NATIVEReward) (NativeFarm.sol#1579)
        - safeNATIVETransfer(msg.sender,pending) (NativeFarm.sol#1638)
                - IERC20(NATIVE).transfer(_to,NATIVEBal) (NativeFarm.sol#1692)
                - IERC20(NATIVE).transfer(_to,_NATIVEAmt) (NativeFarm.sol#1694)
        - sharesRemoved = IStrategy(poolInfo[_pid].strat).withdraw(msg.sender,_wantAmt) (NativeFarm.sol#1647-1648)
        - pool.want.safeTransfer(address(msg.sender),_wantAmt) (NativeFarm.sol#1660)
        Event emitted after the call(s):
        - Withdraw(msg.sender,_pid,_wantAmt) (NativeFarm.sol#1663)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
NativeFarm.add(uint256,IERC20,address) (NativeFarm.sol#1473-1489) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > startTime (NativeFarm.sol#1478)
```

```
NativeFarm.pendingNATIVE(uint256,address) (NativeFarm.sol#1515-1535) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > pool.lastRewardTime && sharesTotal != 0 (NativeFarm.sol#1524)
NativeFarm.updatePool(uint256) (NativeFarm.sol#1556-1585) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp <= pool.lastRewardTime (NativeFarm.sol#1558)
        - multiplier <= 0 (NativeFarm.sol#1567)
NativeFarm.withdraw(uint256,uint256) (NativeFarm.sol#1619-1664) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(user.shares > 0,user.shares is 0) (NativeFarm.sol#1629)
        - require(bool,string)(sharesTotal > 0,sharesTotal is 0) (NativeFarm.sol#1630)
        - _wantAmt > amount (NativeFarm.sol#1643)
        - _wantAmt > 0 (NativeFarm.sol#1646)
        - sharesRemoved > user.shares (NativeFarm.sol#1650)
        - wantBal < _wantAmt (NativeFarm.sol#1657)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (NativeFarm.sol#616-627) uses assembly
        - INLINE ASM (NativeFarm.sol#623-625)
Address._verifyCallResult(bool,bytes,string) (NativeFarm.sol#820-841) uses assembly
        - INLINE ASM (NativeFarm.sol#833-836)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (NativeFarm.sol#677-682) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (NativeFarm.sol#709-721) is never used and should be removed
Address.functionDelegateCall(address,bytes) (NativeFarm.sol#790-800) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (NativeFarm.sol#808-818) is never used and should be removed
Address.functionStaticCall(address,bytes) (NativeFarm.sol#753-764) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (NativeFarm.sol#772-782) is never used and should be removed
Address.sendValue(address,uint256) (NativeFarm.sol#645-657) is never used and should be removed
```

```
Context._msgData() (NativeFarm.sol#14-17) is never used and should be removed
ERC20._burn(address,uint256) (NativeFarm.sol#528-539) is never used and should be removed
ERC20._mint(address,uint256) (NativeFarm.sol#507-515) is never used and should be removed
ERC20._setupDecimals(uint8) (NativeFarm.sol#573-575) is never used and should be removed
EnumerableSet._add(EnumerableSet.Set,bytes32) (NativeFarm.sol#985-995) is never used and should be removed
EnumerableSet._at(EnumerableSet.Set,uint256) (NativeFarm.sol#1066-1076) is never used and should be removed
EnumerableSet._contains(EnumerableSet.Set,bytes32) (NativeFarm.sol#1041-1047) is never used and should be removed
EnumerableSet._length(EnumerableSet.Set) (NativeFarm.sol#1052-1054) is never used and should be removed
EnumerableSet._remove(EnumerableSet.Set,bytes32) (NativeFarm.sol#1003-1036) is never used and should be removed
EnumerableSet.add(EnumerableSet.AddressSet,address) (NativeFarm.sol#1158-1163) is never used and should be removed
EnumerableSet.add(EnumerableSet.Bytes32Set,bytes32) (NativeFarm.sol#1090-1095) is never used and should be removed
EnumerableSet.add(EnumerableSet.UintSet,uint256) (NativeFarm.sol#1226-1228) is never used and should be removed
EnumerableSet.at(EnumerableSet.AddressSet,uint256) (NativeFarm.sol#1206-1212) is never used and should be removed
EnumerableSet.at(EnumerableSet.Bytes32Set,uint256) (NativeFarm.sol#1138-1144) is never used and should be removed
EnumerableSet.at(EnumerableSet.UintSet,uint256) (NativeFarm.sol#1271-1277) is never used and should be removed
EnumerableSet.contains(EnumerableSet.AddressSet,address) (NativeFarm.sol#1181-1187) is never used and should be removed
EnumerableSet.contains(EnumerableSet.Bytes32Set,bytes32) (NativeFarm.sol#1113-1119) is never used and should be removed
EnumerableSet.contains(EnumerableSet.UintSet,uint256) (NativeFarm.sol#1246-1252) is never used and should be removed
EnumerableSet.length(EnumerableSet.AddressSet) (NativeFarm.sol#1192-1194) is never used and should be removed
EnumerableSet.length(EnumerableSet.Bytes32Set) (NativeFarm.sol#1124-1126) is never used and should be removed
EnumerableSet.length(EnumerableSet.UintSet) (NativeFarm.sol#1257-1259) is never used and should be removed
EnumerableSet.remove(EnumerableSet.AddressSet,address) (NativeFarm.sol#1171-1176) is never used and should be removed
EnumerableSet.remove(EnumerableSet.Bytes32Set,bytes32) (NativeFarm.sol#1103-1108) is never used and should be removed
EnumerableSet.remove(EnumerableSet.UintSet,uint256) (NativeFarm.sol#1236-1241) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (NativeFarm.sol#878-895) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (NativeFarm.sol#914-932) is never used and should be removed
SafeMath.mod(uint256,uint256) (NativeFarm.sol#149-151) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (NativeFarm.sol#165-172) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (NativeFarm.sol#645-657):
        - (success) = recipient.call{value: amount}() (NativeFarm.sol#652)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (NativeFarm.sol#729-745):
        - (success,returndata) = target.call{value: value}(data) (NativeFarm.sol#742-743)
Low level call in Address.functionStaticCall(address,bytes,string) (NativeFarm.sol#772-782):
        - (success,returndata) = target.staticcall(data) (NativeFarm.sol#780)
Low level call in Address.functionDelegateCall(address,bytes,string) (NativeFarm.sol#808-818):
        - (success,returndata) = target.delegatecall(data) (NativeFarm.sol#816)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter NativeFarm.add(uint256,IERC20,address)._allocPoint (NativeFarm.sol#1474) is not in mixedCase
Parameter NativeFarm.add(uint256,IERC20,address)._want (NativeFarm.sol#1475) is not in mixedCase
Parameter NativeFarm.add(uint256,IERC20,address)._strat (NativeFarm.sol#1476) is not in mixedCase
Parameter NativeFarm.set(uint256,uint256)._pid (NativeFarm.sol#1493) is not in mixedCase
Parameter NativeFarm.set(uint256,uint256)._allocPoint (NativeFarm.sol#1494) is not in mixedCase
Parameter NativeFarm.getMultiplier(uint256,uint256)._from (NativeFarm.sol#1503) is not in mixedCase
Parameter NativeFarm.getMultiplier(uint256,uint256)._to (NativeFarm.sol#1503) is not in mixedCase
Parameter NativeFarm.pendingNATIVE(uint256,address)._pid (NativeFarm.sol#1515) is not in mixedCase
Parameter NativeFarm.pendingNATIVE(uint256,address)._user (NativeFarm.sol#1515) is not in mixedCase
Parameter NativeFarm.stakedWantTokens(uint256,address)._pid (NativeFarm.sol#1538) is not in mixedCase
Parameter NativeFarm.stakedWantTokens(uint256,address)._user (NativeFarm.sol#1538) is not in mixedCase
Parameter NativeFarm.updatePool(uint256)._pid (NativeFarm.sol#1556) is not in mixedCase
Parameter NativeFarm.deposit(uint256,uint256)._pid (NativeFarm.sol#1588) is not in mixedCase
Parameter NativeFarm.deposit(uint256,uint256)._wantAmt (NativeFarm.sol#1588) is not in mixedCase
Parameter NativeFarm.withdraw(uint256,uint256)._pid (NativeFarm.sol#1619) is not in mixedCase
Parameter NativeFarm.withdraw(uint256,uint256)._wantAmt (NativeFarm.sol#1619) is not in mixedCase
Parameter NativeFarm.withdrawAll(uint256)._pid (NativeFarm.sol#1666) is not in mixedCase
Parameter NativeFarm.emergencyWithdraw(uint256)._pid (NativeFarm.sol#1671) is not in mixedCase
Parameter NativeFarm.safeNATIVETransfer(address,uint256)._to (NativeFarm.sol#1689) is not in mixedCase
Parameter NativeFarm.safeNATIVETransfer(address,uint256)._NATIVEAmt (NativeFarm.sol#1689) is not in mixedCase
Parameter NativeFarm.inCaseTokensGetStuck(address,uint256)._token (NativeFarm.sol#1698) is not in mixedCase
Parameter NativeFarm.inCaseTokensGetStuck(address,uint256)._amount (NativeFarm.sol#1698) is not in mixedCase
Constant NativeFarm.ownerNATIVEReward (NativeFarm.sol#1447) is not in UPPER_CASE_WITH_UNDERSCORES
Constant NativeFarm.NATIVEMaxSupply (NativeFarm.sol#1449) is not in UPPER_CASE_WITH_UNDERSCORES
Constant NativeFarm.NATIVEPerSecond (NativeFarm.sol#1451) is not in UPPER_CASE_WITH_UNDERSCORES
Constant NativeFarm.startTime (NativeFarm.sol#1453) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (NativeFarm.sol#15)" inContext (NativeFarm.sol#9-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
NativeFarm.slitherConstructorConstantVariables() (NativeFarm.sol#1414-1706) uses literals with too many digits:
        - NATIVEPerSecond = 107205733000000000 (NativeFarm.sol#1451)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
NATIVEToken (NativeFarm.sol#1382-1384) does not implement functions:
```

```
NATIVEToken (NativeFarm.sol#1382-1384) does not implement functions:
        - NATIVEToken.mint(address,uint256) (NativeFarm.sol#1383)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (NativeFarm.sol#290-292)
symbol() should be declared external:
        - ERC20.symbol() (NativeFarm.sol#298-300)
decimals() should be declared external:
        - ERC20.decimals() (NativeFarm.sol#315-317)
totalSupply() should be declared external:
        - ERC20.totalSupply() (NativeFarm.sol#322-324)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (NativeFarm.sol#329-331)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (NativeFarm.sol#341-349)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (NativeFarm.sol#354-362)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (NativeFarm.sol#371-379)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (NativeFarm.sol#394-409)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (NativeFarm.sol#423-434)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (NativeFarm.sol#450-464)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (NativeFarm.sol#1319-1322)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (NativeFarm.sol#1328-1335)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:NativeFarm.sol analyzed (12 contracts with 75 detectors), 101 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> StrategyV3_Jetswap.sol

```
INFO:Detectors:
StrategyV3.deposit(address,uint256) (StrategyV3_Jetswap.sol#1118-1154) performs a multiplication on the result of a division:
        -_wantAmt = _wantAmt.mul(depositFeeFactor).div(depositFeeFactorMax) (StrategyV3_Jetswap.sol#1134)
        -sharesAdded = _wantAmt.mul(sharesTotal).mul(entranceFeeFactor).div(wantLockedTotal).div(entranceFeeFactorMax) (StrategyV3_Jetswa
p.sol#1139-1143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in StrategyV3.withdraw(address,uint256) (StrategyV3_Jetswap.sol#1173-1210):
        External calls:
        - _unfarm(_wantAmt) (StrategyV3_Jetswap.sol#1193)
                - IXswapFarm(farmContractAddress).withdraw(pid,_wantAmt) (StrategyV3_Jetswap.sol#1170)
        State variables written after the call(s):
        - wantLockedTotal = wantLockedTotal.sub(_wantAmt) (StrategyV3_Jetswap.sol#1205)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
StrategyV3.earn() (StrategyV3_Jetswap.sol#1219-1309) ignores return value by IXRouter02(uniRouterAddress).addLiquidity(token0Address,toke
n1Address,token0Amt,token1Amt,0,0,address(this),now + routerDeadlineDuration) (StrategyV3_Jetswap.sol#1294-1303)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
StrategyV3.setGov(address) (StrategyV3_Jetswap.sol#1452-1454) should emit an event for:
        - govAddress = _govAddress (StrategyV3_Jetswap.sol#1453)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control
INFO:Detectors:
StrategyV3.setGov(address)._govAddress (StrategyV3_Jetswap.sol#1452) lacks a zero-check on :
                - govAddress = _govAddress (StrategyV3_Jetswap.sol#1453)
StrategyV3.setBuybackRouterAddress(address)._buybackRouterAddress (StrategyV3_Jetswap.sol#1473) lacks a zero-check on :
                - buybackRouterAddress = _buybackRouterAddress (StrategyV3_Jetswap.sol#1474)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in StrategyV3.deposit(address,uint256) (StrategyV3_Jetswap.sol#1118-1154):
        External calls:
        - IERC20(wantAddress).safeTransferFrom(address(msg.sender),address(this),_wantAmt) (StrategyV3_Jetswap.sol#1126-1130)
        State variables written after the call(s):
        - sharesTotal = sharesTotal.add(sharesAdded) (StrategyV3_Jetswap.sol#1145)
        - wantLockedTotal = wantLockedTotal.add(_wantAmt) (StrategyV3_Jetswap.sol#1150)
Reentrancy in StrategyV3.earn() (StrategyV3_Jetswap.sol#1219-1309):
        External calls:
```

```
        - _harvest() (StrategyV3_Jetswap.sol#1223)
                - IXswapFarm(farmContractAddress).withdraw(pid,_wantAmt) (StrategyV3_Jetswap.sol#1170)
        - earnedAmt = distributeFees(earnedAmt) (StrategyV3_Jetswap.sol#1228)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (StrategyV3_Jetswap.sol#606-610)
                - IERC20(earnedAddress).safeTransfer(govAddress,fee) (StrategyV3_Jetswap.sol#1380)
                - (success,returndata) = target.call{value: value}(data) (StrategyV3_Jetswap.sol#403-404)
        - earnedAmt = buyBack(earnedAmt) (StrategyV3_Jetswap.sol#1229)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (StrategyV3_Jetswap.sol#606-610)
                - IXRouter02(_uniRouterAddress).swapExactTokensForTokensSupportingFeeOnTransferTokens(_amountIn,amountOut.mul(_slippageFa
ctor).div(1000),_path,_to,_deadline) (StrategyV3_Jetswap.sol#1507-1514)
                - (success,returndata) = target.call{value: value}(data) (StrategyV3_Jetswap.sol#403-404)
                - IERC20(earnedAddress).safeIncreaseAllowance(uniRouterAddress,buyBackAmt) (StrategyV3_Jetswap.sol#1323-1326)
                - IXRouter02(uniRouterAddress).swapExactTokensForTokensSupportingFeeOnTransferTokens(buyBackAmt,0,earnedToWFTMPath,addres
s(this),now + routerDeadlineDuration) (StrategyV3_Jetswap.sol#1328-1335)
                - IERC20(wftmAddress).safeIncreaseAllowance(buybackRouterAddress,wftmAmt) (StrategyV3_Jetswap.sol#1341-1344)
                - IXRouter02(buybackRouterAddress).swapExactTokensForTokensSupportingFeeOnTransferTokens(wftmAmt,0,WFTMToNATIVEPath,buyBa
ckAddress,now + routerDeadlineDuration) (StrategyV3_Jetswap.sol#1346-1353)
                - IERC20(earnedAddress).safeIncreaseAllowance(uniRouterAddress,buyBackAmt) (StrategyV3_Jetswap.sol#1356-1359)
        - IERC20(earnedAddress).safeIncreaseAllowance(uniRouterAddress,earnedAmt) (StrategyV3_Jetswap.sol#1233-1236)
        - _safeSwap(uniRouterAddress,earnedAmt,slippageFactor,earnedToToken0Path,address(this),now + routerDeadlineDuration) (StrategyV3_
Jetswap.sol#1239-1246)
                - IXRouter02(_uniRouterAddress).swapExactTokensForTokensSupportingFeeOnTransferTokens(_amountIn,amountOut.mul(_slippageFa
ctor).div(1000),_path,_to,_deadline) (StrategyV3_Jetswap.sol#1507-1514)
        External calls sending eth:
        - earnedAmt = distributeFees(earnedAmt) (StrategyV3_Jetswap.sol#1228)
                - (success,returndata) = target.call{value: value}(data) (StrategyV3_Jetswap.sol#403-404)
        - earnedAmt = buyBack(earnedAmt) (StrategyV3_Jetswap.sol#1229)
                - (success,returndata) = target.call{value: value}(data) (StrategyV3_Jetswap.sol#403-404)
        State variables written after the call(s):
        - lastEarnTime = block.timestamp (StrategyV3_Jetswap.sol#1248)
Reentrancy in StrategyV3.earn() (StrategyV3_Jetswap.sol#1219-1309):
        External calls:
        - _harvest() (StrategyV3_Jetswap.sol#1223)
                - IXswapFarm(farmContractAddress).withdraw(pid,_wantAmt) (StrategyV3_Jetswap.sol#1170)
        - earnedAmt = distributeFees(earnedAmt) (StrategyV3_Jetswap.sol#1228)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (StrategyV3_Jetswap.sol#606-610)
                - IERC20(earnedAddress).safeTransfer(govAddress,fee) (StrategyV3_Jetswap.sol#1380)
                - (success,returndata) = target.call{value: value}(data) (StrategyV3_Jetswap.sol#403-404)
```

```
        State variables written after the call(s):
        - lastEarnTime = block.timestamp (StrategyV3_Jetswap.sol#1248)
Reentrancy in StrategyV3.earn() (StrategyV3_Jetswap.sol#1219-1309):
        External calls:
        - _harvest() (StrategyV3_Jetswap.sol#1223)
                - IXswapFarm(farmContractAddress).withdraw(pid,_wantAmt) (StrategyV3_Jetswap.sol#1170)
        - earnedAmt = distributeFees(earnedAmt) (StrategyV3_Jetswap.sol#1228)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (StrategyV3_Jetswap.sol#606-610)
                - IERC20(earnedAddress).safeTransfer(govAddress,fee) (StrategyV3_Jetswap.sol#1380)
                - (success,returndata) = target.call{value: value}(data) (StrategyV3_Jetswap.sol#403-404)
        - earnedAmt = buyBack(earnedAmt) (StrategyV3_Jetswap.sol#1229)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (StrategyV3_Jetswap.sol#606-610)
                - IXRouter02(_uniRouterAddress).swapExactTokensForTokensSupportingFeeOnTransferTokens(_amountIn,amountOut.mul(_slippageFa
ctor).div(1000),_path,_to,_deadline) (StrategyV3_Jetswap.sol#1507-1514)
                - (success,returndata) = target.call{value: value}(data) (StrategyV3_Jetswap.sol#403-404)
                - IERC20(earnedAddress).safeIncreaseAllowance(uniRouterAddress,buyBackAmt) (StrategyV3_Jetswap.sol#1323-1326)
                - IXRouter02(uniRouterAddress).swapExactTokensForTokensSupportingFeeOnTransferTokens(buyBackAmt,0,earnedToWFTMPath,addres
s(this),now + routerDeadlineDuration) (StrategyV3_Jetswap.sol#1328-1335)
                - IERC20(wftmAddress).safeIncreaseAllowance(buybackRouterAddress,wftmAmt) (StrategyV3_Jetswap.sol#1341-1344)
                - IXRouter02(buybackRouterAddress).swapExactTokensForTokensSupportingFeeOnTransferTokens(wftmAmt,0,WFTMToNATIVEPath,buyBa
ckAddress,now + routerDeadlineDuration) (StrategyV3_Jetswap.sol#1346-1353)
                - IERC20(earnedAddress).safeIncreaseAllowance(uniRouterAddress,buyBackAmt) (StrategyV3_Jetswap.sol#1356-1359)
        - IERC20(earnedAddress).safeIncreaseAllowance(uniRouterAddress,earnedAmt) (StrategyV3_Jetswap.sol#1253-1256)
        - _safeSwap(uniRouterAddress,earnedAmt.div(2),slippageFactor,earnedToToken0Path,address(this),now + routerDeadlineDuration) (Stra
tegyV3_Jetswap.sol#1260-1267)
                - IXRouter02(_uniRouterAddress).swapExactTokensForTokensSupportingFeeOnTransferTokens(_amountIn,amountOut.mul(_slippageFa
ctor).div(1000),_path,_to,_deadline) (StrategyV3_Jetswap.sol#1507-1514)
        - _safeSwap(uniRouterAddress,earnedAmt.div(2),slippageFactor,earnedToToken1Path,address(this),now + routerDeadlineDuration) (Stra
tegyV3_Jetswap.sol#1272-1279)
```

```
                - IXRouter02(_uniRouterAddress).swapExactTokensForTokensSupportingFeeOnTransferTokens(_amountIn,amountOut.mul(_slippageFa
ctor).div(1000),_path,_to,_deadline) (StrategyV3_Jetswap.sol#1507-1514)
        - _safeSwap(uniRouterAddress,earnedAmt.div(2),slippageFactor,earnedToToken1Path,address(this),now + routerDeadlineDuration) (Stra
tegyV3_Jetswap.sol#1272-1279)
                - IXRouter02(_uniRouterAddress).swapExactTokensForTokensSupportingFeeOnTransferTokens(_amountIn,amountOut.mul(_slippageFa
ctor).div(1000),_path,_to,_deadline) (StrategyV3_Jetswap.sol#1507-1514)
        - IERC20(token0Address).safeIncreaseAllowance(uniRouterAddress,token0Amt) (StrategyV3_Jetswap.sol#1286-1289)
        - IERC20(token1Address).safeIncreaseAllowance(uniRouterAddress,token1Amt) (StrategyV3_Jetswap.sol#1290-1293)
        - IXRouter02(uniRouterAddress).addLiquidity(token0Address,token1Address,token0Amt,token1Amt,0,0,address(this),now + routerDeadlin
eDuration) (StrategyV3_Jetswap.sol#1294-1303)
        External calls sending eth:
        - earnedAmt = distributeFees(earnedAmt) (StrategyV3_Jetswap.sol#1228)
                - (success,returndata) = target.call{value: value}(data) (StrategyV3_Jetswap.sol#403-404)
        - earnedAmt = buyBack(earnedAmt) (StrategyV3_Jetswap.sol#1229)
                - (success,returndata) = target.call{value: value}(data) (StrategyV3_Jetswap.sol#403-404)
        State variables written after the call(s):
        - lastEarnTime = block.timestamp (StrategyV3_Jetswap.sol#1306)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Address.isContract(address) (StrategyV3_Jetswap.sol#263-274) uses assembly
        - INLINE ASM (StrategyV3_Jetswap.sol#270-272)
Address._verifyCallResult(bool,bytes,string) (StrategyV3_Jetswap.sol#481-502) uses assembly
        - INLINE ASM (StrategyV3_Jetswap.sol#494-497)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (StrategyV3_Jetswap.sol#338-343) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (StrategyV3_Jetswap.sol#370-382) is never used and should be removed
Address.functionDelegateCall(address,bytes) (StrategyV3_Jetswap.sol#451-461) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (StrategyV3_Jetswap.sol#469-479) is never used and should be removed
Address.functionStaticCall(address,bytes) (StrategyV3_Jetswap.sol#414-425) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (StrategyV3_Jetswap.sol#433-443) is never used and should be removed
Address.sendValue(address,uint256) (StrategyV3_Jetswap.sol#306-318) is never used and should be removed
Address.toPayable(address) (StrategyV3_Jetswap.sol#282-288) is never used and should be removed
Context._msgData() (StrategyV3_Jetswap.sol#912-915) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (StrategyV3_Jetswap.sol#539-556) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (StrategyV3_Jetswap.sol#575-593) is never used and should be removed
SafeMath.mod(uint256,uint256) (StrategyV3_Jetswap.sol#219-221) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (StrategyV3_Jetswap.sol#235-242) is never used and should be removed
```

```
StrategyV3._wrapFTM() (StrategyV3_Jetswap.sol#1487-1493) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (StrategyV3_Jetswap.sol#306-318):
        - (success) = recipient.call{value: amount}() (StrategyV3_Jetswap.sol#313)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (StrategyV3_Jetswap.sol#390-406):
        - (success,returndata) = target.call{value: value}(data) (StrategyV3_Jetswap.sol#403-404)
Low level call in Address.functionStaticCall(address,bytes,string) (StrategyV3_Jetswap.sol#433-443):
        - (success,returndata) = target.staticcall(data) (StrategyV3_Jetswap.sol#441)
Low level call in Address.functionDelegateCall(address,bytes,string) (StrategyV3_Jetswap.sol#469-479):
        - (success,returndata) = target.delegatecall(data) (StrategyV3_Jetswap.sol#477)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IXRouter01.WETH() (StrategyV3_Jetswap.sol#650) is not in mixedCase
Parameter StrategyV3.deposit(address,uint256)._wantAmt (StrategyV3_Jetswap.sol#1118) is not in mixedCase
Parameter StrategyV3.withdraw(address,uint256)._wantAmt (StrategyV3_Jetswap.sol#1173) is not in mixedCase
Parameter StrategyV3.buyBack(uint256)._earnedAmt (StrategyV3_Jetswap.sol#1311) is not in mixedCase
Parameter StrategyV3.distributeFees(uint256)._earnedAmt (StrategyV3_Jetswap.sol#1374) is not in mixedCase
Parameter StrategyV3.setEntranceFeeFactor(uint256)._entranceFeeFactor (StrategyV3_Jetswap.sol#1441) is not in mixedCase
Parameter StrategyV3.setControllerFee(uint256)._controllerFee (StrategyV3_Jetswap.sol#1447) is not in mixedCase
Parameter StrategyV3.setGov(address)._govAddress (StrategyV3_Jetswap.sol#1452) is not in mixedCase
Parameter StrategyV3.setDepositFeeFactor(uint256)._depositFeeFactor (StrategyV3_Jetswap.sol#1456) is not in mixedCase
Parameter StrategyV3.setWithdrawFeeFactor(uint256)._withdrawFeeFactor (StrategyV3_Jetswap.sol#1462) is not in mixedCase
Parameter StrategyV3.setbuyBackRate(uint256)._buyBackRate (StrategyV3_Jetswap.sol#1468) is not in mixedCase
Parameter StrategyV3.setBuybackRouterAddress(address)._buybackRouterAddress (StrategyV3_Jetswap.sol#1473) is not in mixedCase
Parameter StrategyV3.inCaseTokensGetStuck(address,uint256,address)._token (StrategyV3_Jetswap.sol#1478) is not in mixedCase
Parameter StrategyV3.inCaseTokensGetStuck(address,uint256,address)._amount (StrategyV3_Jetswap.sol#1479) is not in mixedCase
Parameter StrategyV3.inCaseTokensGetStuck(address,uint256,address)._to (StrategyV3_Jetswap.sol#1480) is not in mixedCase
Variable StrategyV3.NATIVEAddress (StrategyV3_Jetswap.sol#1073) is not in mixedCase
Constant StrategyV3.controllerFeeMax (StrategyV3_Jetswap.sol#1081) is not in UPPER_CASE_WITH_UNDERSCORES
Constant StrategyV3.controllerFeeUL (StrategyV3_Jetswap.sol#1082) is not in UPPER_CASE_WITH_UNDERSCORES
Constant StrategyV3.buyBackRateMax (StrategyV3_Jetswap.sol#1085) is not in UPPER_CASE_WITH_UNDERSCORES
Constant StrategyV3.buyBackRateUL (StrategyV3_Jetswap.sol#1086) is not in UPPER_CASE_WITH_UNDERSCORES
Constant StrategyV3.buyBackAddress (StrategyV3_Jetswap.sol#1087) is not in UPPER_CASE_WITH_UNDERSCORES
Constant StrategyV3.entranceFeeFactorMax (StrategyV3_Jetswap.sol#1090) is not in UPPER_CASE_WITH_UNDERSCORES
Constant StrategyV3.entranceFeeFactorLL (StrategyV3_Jetswap.sol#1091) is not in UPPER_CASE_WITH_UNDERSCORES
Constant StrategyV3.depositFeeFactorMax (StrategyV3_Jetswap.sol#1094) is not in UPPER_CASE_WITH_UNDERSCORES
Constant StrategyV3.depositFeeFactorLL (StrategyV3_Jetswap.sol#1095) is not in UPPER_CASE_WITH_UNDERSCORES
```

```
INFO:Detectors:
Redundant expression "this (StrategyV3_Jetswap.sol#913)" inContext (StrategyV3_Jetswap.sol#907-916)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable IXRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (StrategyV3_Jetswap.sol#
655) is too similar to IXRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (StrategyV
3_Jetswap.sol#656)
Variable StrategyV3.earnedToToken0Path (StrategyV3_Jetswap.sol#1105) is too similar to StrategyV3.earnedToToken1Path (StrategyV3_Jetswap.
sol#1106)
Variable StrategyV3.token0Address (StrategyV3_Jetswap.sol#1064) is too similar to StrategyV3.token1Address (StrategyV3_Jetswap.sol#1065)
Variable StrategyV3.token0ToEarnedPath (StrategyV3_Jetswap.sol#1107) is too similar to StrategyV3.token1ToEarnedPath (StrategyV3_Jetswap.
sol#1108)
Variable StrategyV3_Jetswap.constructor(address[],address[],bool,bool,uint256,address[],address[],address[],address[],address[],address[]
,uint256,uint256,uint256)._earnedToToken0Path (StrategyV3_Jetswap.sol#1532) is too similar to StrategyV3_Jetswap.constructor(address[],ad
dress[],bool,bool,uint256,address[],address[],address[],address[],address[],address[],uint256,uint256,uint256)._earnedToToken1Path (Strat
egyV3_Jetswap.sol#1533)
Variable StrategyV3_Jetswap.constructor(address[],address[],bool,bool,uint256,address[],address[],address[],address[],address[],address[]
,uint256,uint256,uint256)._token0ToEarnedPath (StrategyV3_Jetswap.sol#1534) is too similar to StrategyV3_Jetswap.constructor(address[],ad
dress[],bool,bool,uint256,address[],address[],address[],address[],address[],address[],uint256,uint256,uint256)._token1ToEarnedPath (Strat
egyV3_Jetswap.sol#1535)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
StrategyV3_Jetswap.slitherConstructorConstantVariables() (StrategyV3_Jetswap.sol#1521-1596) uses literals with too many digits:
        - buyBackAddress = 0x000000000000000000000000000000000000dEaD (StrategyV3_Jetswap.sol#1087)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
StrategyV3.routerDeadlineDuration (StrategyV3_Jetswap.sol#1069) should be constant
StrategyV3.slippageFactor (StrategyV3_Jetswap.sol#1101) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
paused() should be declared external:
        - Pausable.paused() (StrategyV3_Jetswap.sol#941-943)
owner() should be declared external:
        - Ownable.owner() (StrategyV3_Jetswap.sol#1014-1016)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (StrategyV3_Jetswap.sol#1033-1036)
farm() should be declared external:
        - StrategyV3.farm() (StrategyV3_Jetswap.sol#1156-1158)
```

```
farm() should be declared external:
        - StrategyV3.farm() (StrategyV3_Jetswap.sol#1156-1158)
earn() should be declared external:
        - StrategyV3.earn() (StrategyV3_Jetswap.sol#1219-1309)
convertDustToEarned() should be declared external:
        - StrategyV3.convertDustToEarned() (StrategyV3_Jetswap.sol#1388-1431)
pause() should be declared external:
        - StrategyV3.pause() (StrategyV3_Jetswap.sol#1433-1435)
setEntranceFeeFactor(uint256) should be declared external:
        - StrategyV3.setEntranceFeeFactor(uint256) (StrategyV3_Jetswap.sol#1441-1445)
setControllerFee(uint256) should be declared external:
        - StrategyV3.setControllerFee(uint256) (StrategyV3_Jetswap.sol#1447-1450)
setGov(address) should be declared external:
        - StrategyV3.setGov(address) (StrategyV3_Jetswap.sol#1452-1454)
setDepositFeeFactor(uint256) should be declared external:
        - StrategyV3.setDepositFeeFactor(uint256) (StrategyV3_Jetswap.sol#1456-1460)
setWithdrawFeeFactor(uint256) should be declared external:
        - StrategyV3.setWithdrawFeeFactor(uint256) (StrategyV3_Jetswap.sol#1462-1466)
setbuyBackRate(uint256) should be declared external:
        - StrategyV3.setbuyBackRate(uint256) (StrategyV3_Jetswap.sol#1468-1471)
setBuybackRouterAddress(address) should be declared external:
        - StrategyV3.setBuybackRouterAddress(address) (StrategyV3_Jetswap.sol#1473-1475)
inCaseTokensGetStuck(address,uint256,address) should be declared external:
        - StrategyV3.inCaseTokensGetStuck(address,uint256,address) (StrategyV3_Jetswap.sol#1477-1485)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:StrategyV3_Jetswap.sol analyzed (15 contracts with 75 detectors), 86 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity static analysis

FAST.sol

Gas & Economy

Gas costs:

Gas requirement of function ERC20.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 290:4:

Gas costs:

Gas requirement of function FAST.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 290:4:

Gas costs:

Gas requirement of function ERC20.symbol is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 298:4:

Gas costs:

Gas requirement of function FAST.symbol is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 298:4:

Gas costs:

Gas requirement of function ERC20.transfer is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 341:4:

Gas costs:

Gas requirement of function FAST.transfer is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 341:4:

Gas costs:

Gas requirement of function ERC20.approve is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 371:4:

Miscellaneous

Constant/View/Pure functions:

SafeMath.sub(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 48:4:

Constant/View/Pure functions:

SafeMath.div(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 109:4:

Constant/View/Pure functions:

SafeMath.mod(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 149:4:

Constant/View/Pure functions:

ERC20._beforeTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 588:4:

Similar variable names:

ERC20.(string,string) : Variables have very similar names "_name" and "name_". Note: Modifiers are currently not considered by this static analysis.
Pos: 282:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 505:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 526:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 556:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 557:8:

NativeFarm.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 729:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeApprove(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 878:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeIncreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 897:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeDecreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 914:4:

Gas & Economy

Gas costs:

Gas requirement of function ERC20.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 290:4:

Gas costs:

Gas requirement of function ERC20.symbol is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 298:4:

Gas costs:

Gas requirement of function ERC20.transfer is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 341:4:

Gas costs:

Gas requirement of function ERC20.approve is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 371:4:

Miscellaneous

Constant/View/Pure functions:

SafeMath.sub(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 48:4:

Constant/View/Pure functions:

SafeMath.div(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 109:4:

Constant/View/Pure functions:

SafeMath.mod(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 149:4:

Constant/View/Pure functions:

ERC20._beforeTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 591:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1629:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1630:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1702:8:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 1030:12:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1329:8:

StrategyV3_Jetswap.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 390:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeApprove(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 539:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeIncreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 558:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeDecreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 575:4:

Block timestamp:

Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp".
"block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 1334:20:

Block timestamp:

Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp".
"block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 1352:20:

Block timestamp:

Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp".
"block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 1409:16:

Block timestamp:

Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp".
"block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 1428:16:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1306:23:

Gas & Economy

Gas costs:

Gas requirement of function StrategyV3_Jetswap.transferOwnership is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 1042:4:

Gas costs:

Gas requirement of function StrategyV3_Jetswap.deposit is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 1118:4:

Gas costs:

Gas requirement of function StrategyV3_Jetswap.farm is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 1156:4:

Gas costs:

Gas requirement of function StrategyV3_Jetswap.withdraw is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 1173:4:

Miscellaneous

Constant/View/Pure functions:

SafeMath.sub(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 118:4:

Constant/View/Pure functions:

SafeMath.div(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 179:4:

Constant/View/Pure functions:

SafeMath.mod(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 219:4:

Constant/View/Pure functions:

Address.isContract(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 263:4:

Solhint Linter

FAST.sol

```
FAST.sol:7:1: Error: Compiler version ^0.6.12 does not satisfy the r
semver requirement
FAST.sol:592:24: Error: Code contains empty blocks
```

NativeFarm.sol

```
NativeFarm.sol:7:1: Error: Compiler version 0.6.12 does not satisfy the
r semver requirement
NativeFarm.sol:595:24: Error: Code contains empty blocks
NativeFarm.sol:743:13: Error: Avoid to use low level calls.
NativeFarm.sol:1447:29: Error: Constant name must be in capitalized
SNAKE_CASE
NativeFarm.sol:1449:29: Error: Constant name must be in capitalized
SNAKE_CASE
NativeFarm.sol:1451:29: Error: Constant name must be in capitalized
SNAKE_CASE
NativeFarm.sol:1453:29: Error: Constant name must be in capitalized
SNAKE_CASE
NativeFarm.sol:1478:34: Error: Avoid to make time-based decisions in
your business logic
NativeFarm.sol:1478:64: Error: Avoid to make time-based decisions in
your business logic
NativeFarm.sol:1524:13: Error: Avoid to make time-based decisions in
your business logic
NativeFarm.sol:1525:69: Error: Avoid to make time-based decisions in
your business logic
NativeFarm.sol:1526:13: Error: Variable name must be in mixedCase
NativeFarm.sol:1558:13: Error: Avoid to make time-based decisions in
your business logic
NativeFarm.sol:1563:35: Error: Avoid to make time-based decisions in
your business logic
NativeFarm.sol:1566:65: Error: Avoid to make time-based decisions in
your business logic
NativeFarm.sol:1570:9: Error: Variable name must be in mixedCase
NativeFarm.sol:1584:31: Error: Avoid to make time-based decisions in
your business logic
NativeFarm.sol:1689:46: Error: Variable name must be in mixedCase
NativeFarm.sol:1690:9: Error: Variable name must be in mixedCase
```

StrategyV3_Jetswap.sol

```
StrategyV3_Jetswap.sol:3:1: Error: Compiler version 0.6.12 does not
satisfy the r semver requirement
StrategyV3_Jetswap.sol:404:13: Error: Avoid using low level calls.
StrategyV3_Jetswap.sol:650:5: Error: Function name must be in mixedCase
StrategyV3_Jetswap.sol:1052:1: Error: Contract has 31 states
declarations but allowed no more than 15
StrategyV3_Jetswap.sol:1073:20: Error: Variable name must be in
mixedCase
StrategyV3_Jetswap.sol:1081:29: Error: Constant name must be in
capitalized SNAKE_CASE
StrategyV3_Jetswap.sol:1082:29: Error: Constant name must be in
capitalized SNAKE_CASE
StrategyV3_Jetswap.sol:1085:29: Error: Constant name must be in
capitalized SNAKE_CASE
StrategyV3_Jetswap.sol:1086:29: Error: Constant name must be in
capitalized SNAKE_CASE
StrategyV3_Jetswap.sol:1087:29: Error: Constant name must be in
capitalized SNAKE_CASE
StrategyV3_Jetswap.sol:1090:29: Error: Constant name must be in
capitalized SNAKE_CASE
StrategyV3_Jetswap.sol:1091:29: Error: Constant name must be in
capitalized SNAKE_CASE
StrategyV3_Jetswap.sol:1094:29: Error: Constant name must be in
capitalized SNAKE_CASE
StrategyV3_Jetswap.sol:1095:29: Error: Constant name must be in
capitalized SNAKE_CASE
StrategyV3_Jetswap.sol:1098:29: Error: Constant name must be in
capitalized SNAKE_CASE
StrategyV3_Jetswap.sol:1099:29: Error: Constant name must be in
capitalized SNAKE_CASE
StrategyV3_Jetswap.sol:1102:29: Error: Constant name must be in
capitalized SNAKE_CASE
StrategyV3_Jetswap.sol:1110:22: Error: Variable name must be in
mixedCase
StrategyV3_Jetswap.sol:1118:22: Error: Variable "_userAddress" is
unused
StrategyV3_Jetswap.sol:1173:23: Error: Variable "_userAddress" is
unused
StrategyV3_Jetswap.sol:1245:21: Error: Avoid to make time-based
decisions in your business logic
StrategyV3_Jetswap.sol:1248:28: Error: Avoid to make time-based
decisions in your business logic
StrategyV3_Jetswap.sol:1266:17: Error: Avoid to make time-based
decisions in your business logic
StrategyV3_Jetswap.sol:1278:17: Error: Avoid to make time-based
decisions in your business logic
StrategyV3_Jetswap.sol:1302:17: Error: Avoid to make time-based
decisions in your business logic
StrategyV3_Jetswap.sol:1306:24: Error: Avoid to make time-based
decisions in your business logic
StrategyV3_Jetswap.sol:1334:21: Error: Avoid to make time-based
decisions in your business logic
StrategyV3_Jetswap.sol:1352:21: Error: Avoid to make time-based
decisions in your business logic
StrategyV3_Jetswap.sol:1367:17: Error: Avoid to make time-based
```

```
decisions in your business logic
StrategyV3_Jetswap.sol:1409:17: Error: Avoid to make time-based
decisions in your business logic
StrategyV3_Jetswap.sol:1428:17: Error: Avoid to make time-based
decisions in your business logic
StrategyV3_Jetswap.sol:1521:1: Error: Contract name must be in
CamelCase
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Email: audit@EtherAuthority.io