DEFIYIELD

TECHNICAL ANALYSIS FOR TETU.IO

JULY 28, 2021

CONTENT

1. ABOUT DEFIYIELD

DeFiYield is one of the leading smart contract auditing providers focused on checking security of yield farming projects and the world's only DeFi cross-chain asset management protocol based on machine learning techniques.

Our first audits were conducted back in July 2020, shortly after the yield farming industry boomed, bringing impressive return opportunities for users. At the same time, scams happened every day and users were not protected against them in any way. No one performed yield-farming-focused audits at the time and a lot of projects were launching without even doing proper internal audits. This is why DeFiYield took the lead and has been developing and pushing security standards in the community since then.

2. PROJECT SUMMARY

Project Name	Tetu.io
Blockchain	Polygon (Matic)
Language	Solidity
Scope	Tetu.io is a DeFi application developed on Polygon. The project provides automated yield farming strategies and uses the native governance token – TETU.

3. EXECUTIVE SUMMARY

Eleven smart contracts were analysed to check the availability of code vulnerabilities:

Solidity

- Bookkeeper.sol
- Controller.sol
- FeeRewardForwarder.sol
- MCv2StrategyFullBuyback.sol
- MintHelper.sol
- NoopStrategy.sol
- NotifyHelper.sol
- RewardToken.sol
- SNXStrategyFullBuyback.sol
- SmartVault.sol
- WaultStrategyFullBuyback.sol

Total Issues Found: 71

Severity	Count	Title
Critical	10	- A privileged EOA can change protocol parameters - A privileged EOA can call a function that allows to withdraw all funds located in the contract to a needed address
High	0	-
Medium	3	- Unchecked return value
Low	14	- Incorrect Solidity version - State variables that should be declared constant - A privileged EOA can call a function that allows to withdraw all funds located in the contract to a needed address - A privileged EOA can change address of token reward distribution
Info	44	- Public function that should be declared external - Comparison to boolean constant

4. METHODOLOGY

4.1 Smart Contract Code Analysis

4.1.1 Manual Check

DefiYield's system for manual smart contract code auditing is based on experience from analyzing hundreds of malicious and vulnerable smart contracts. The system allows the DefiYield Safe auditors to consistently go through all smart contract elements and their combinations most frequently used to steal user funds.

Issues covered:

Unlimited minting to a malicious destination

Infinite token supply

Dangerous token migration

Pausing token transfers anytime for unlimited period

Pausing token transfer for limited period (defined in the contract)

Pausing funds withdrawals (Centralized pausing for any funds withdrawals)

Pausing funds withdrawals with emergency withdrawal available

Proxy patterns

Funds lock with centralized control

Unfair token distribution: high % of team rewards

Suspicious functions

Insufficient timelock for important contract changes

Overprivileged EOA-contract-owner

a. The owner can call a function that allows to withdraw all staked in the contract funds to a needed address;
b. The owner can change address of token reward distribution;
c. The owner can change location of staked user funds;
Unrestricted fee setting
a. withdrawal fee can be set up to 100%;
b. user reward fee can be decreased;
c. Team reward increased without any limitations in centralized way;
d. Other protocol fees with unexpected security consequences)

Using a singular exchange as a price source.

4.1.2 Automated Check

Safe by DeFiYield

Safe is a machine learning code scanner designed by DeFiYield for automated smart contract security checks. It focuses on detection of DeFi-specific smart contract vulnerabilities and malicious functions that are the most frequent reasons of rug pulls and hacker attacks.

Technique applied: syntax tree code representation checked against bug patterns.

Issues covered:

1. Unverified contracts unlimited minting to a malicious destination
2. dangerous token migration
3. pausing token transfers anytime for unlimited period
4. pausing token transfer for limited period (defined in the contract)
5. pausing funds withdrawals (centralized pausing for any funds withdrawals)
6. pausing funds withdrawals with emergency withdrawal available
7. proxy patterns
8. funds lock with centralized control.

MythX

Technique applied: Symbolic execution.

Issues covered:

1. Assert Violation
2. Integer Overflow and Underflow
3. Arbitrary Jump with Function Type Variable
4. Write to Arbitrary Storage Location
5. Uninitialized Storage Pointer
6. Outdated Compiler Version
7. Floating Pragma, Unchecked Call Return Value
8. Unprotected Ether Withdrawal
9. Unprotected SELFDESTRUCT Instruction
10. Reentrancy, State Variable Default Visibility
11. Uninitialized Storage Pointer
12. Use of Deprecated Solidity Functions
13. Delegatecall to Untrusted Callee
14. DoS with Failed Call
15. Authorization through tx.origin
16. Block values as a proxy for time

17. Incorrect Constructor Name
18. Shadowing State Variables
19. Weak Sources of Randomness from Chain Attributes
20. Requirement Violation
21. Write to Arbitrary Storage Location
22. DoS With Block Gas Limit
23. Typographical Error
24. Right-To-Left-Override control character (U+202E

Slither

Technique applied: Symbolic execution.

Issues covered:

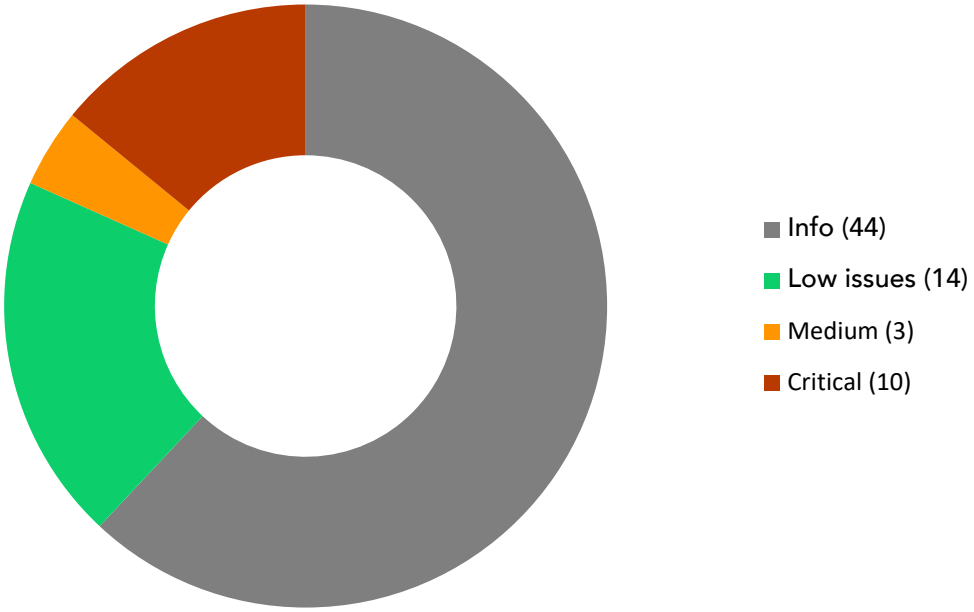
1. Modifying storage array by value
2. The order of parameters in a shift instruction is incorrect
3. Multiple constructor schemes
4. Contract's name reused
5. Public mappings with nested variables
6. Right-To-Left-Override control character is used
7. State variables shadowing
8. Functions allowing anyone to destruct the contract
9. Uninitialized state variables
10. Uninitialized storage variables
11. Unprotected upgradeable contract
12. Functions that send Ether to arbitrary destination
13. Tainted array length assignment
14. Controlled delegatecall destination
15. Reentrancy vulnerabilities (theft of ethers)
16. Signed storage integer array compiler bug
17. Unchecked tokens transfer
18. Weak PRNG
19. Detect dangerous enum conversion
20. Incorrect ERC20 interfaces
21. Incorrect ERC721 interfaces
22. Dangerous strict equalities

23. Contracts that lock ether
24. Deletion on mapping containing a structure
25. State variables shadowing from abstract contracts
26. Tautology or contradiction
27. Unused write
28. Misuse of Boolean constant
29. Constant functions using assembly code
30. Constant functions changing the state
31. Imprecise arithmetic operations order
32. Reentrancy vulnerabilities (no theft of ethers)
33. Reused base constructor
34. Dangerous usage of tx.origin
35. Unchecked low-level calls
36. Unchecked send
37. Uninitialized local variables
38. Unused return values
39. Modifiers that can return the default value
40. Built-in symbol shadowing; Local variables shadowing
41. Uninitialized function pointer calls in constructors
42. Local variables used prior their declaration
43. Constructor called not implemented
44. Multiple calls in a loop
45. Missing Events Access Control
46. Missing Events Arithmetic
47. Dangerous unary expressions
48. Missing Zero Address Validation
49. Benign reentrancy vulnerabilities
50. Reentrancy vulnerabilities leading to out-of-order Events
51. Dangerous usage of block.timestamp
52. Assembly usage
53. Assert state change
54. Comparison to boolean constant
55. Deprecated Solidity Standards
56. Un-indexed ERC20 event parameters
57. Function initializing state variables
58. Low level calls
59. Missing inheritance
60. Conformity to Solidity naming conventions
61. If different pragma directives are used
62. Redundant statements
63. Incorrect Solidity version
64. Unimplemented functions
65. Unused state variables

66. Costly operations in a loop
67. Functions that are not used
68. Reentrancy vulnerabilities through send and transfer
69. Variable names are too similar
70. Conformance to numeric notation best practices
71. State variables that could be declared constant
72. Public function that could be declared externally.

4.1.3 Issue Classification by Severity

Critical	Issues that can directly cause a loss of underlying funds with high probability. These issues must be removed ASAP.
High	There is a possibility of negative impacts on funds managed by the smart contract when certain conditions come into action.
Medium	Issues that affect contract functionality without causing financial losses, must be addressed by the developers.
Low	The issues must be addressed to follow the best SC coding practice.
Info	The issues refer to the best SC coding practice and don't cause any problems with using SCs. Their handling depends on the decision of the dev team.



Total Issues: 71

Related Smart Contract	ID	Issue name	Category	Severity	Status
Bookkeeper.sol	177	Incorrect Solidity version	Solidity Coding Best Practices	Low	✔ Resolved
	184	Public function that should be declared external (x2)	Solidity Coding Best Practices	Info	✔ Resolved
Controller.sol	184	Public function that should be declared external	Solidity Coding Best Practices	Info	✔ Resolved
	198-d	A privileged EOA can change protocol parameters (x7)	DeFi-Specific	Critical	✔ Resolved
	198-a	A privileged EOA can call a function that allows to withdraw funds to a needed address	DeFi-Specific	Low	Acknowledged
	198-b	A privileged EOA can change address of token reward distribution	DeFi-Specific	Low	Acknowledged
	177	Incorrect Solidity version	Solidity Coding Best Practices	Low	✔ Resolved
	170	Comparison to boolean constant	Solidity Coding Best Practices	Info	✔ Resolved

FeeRewardForwarder.sol	184	Public function that should be declared external	Solidity Coding Best Practices	Info	✔ Resolved
	177	Incorrect Solidity version	Solidity Coding Best Practices	Low	✔ Resolved
	104-b	Unchecked return value	Control Flow	Medium	✔ Resolved
MCv2StrategyFullBuyback.sol	184	Public function that should be declared external (x6)	Solidity Coding Best Practices	Info	✔ Resolved
	177	Incorrect Solidity version	Solidity Coding Best Practices	Low	✔ Resolved
MintHelper.sol	183	State variables that should be declared constant	Solidity Coding Best Practices	Low	✔ Resolved
	198-d	A privileged EOA can change protocol parameters	DeFi-Specific	Critical	✔ Resolved
	177	Incorrect Solidity version	Solidity Coding Best Practices	Low	✔ Resolved
	184	Public function that should be declared external	Solidity Coding Best Practices	Info	✔ Resolved
NoopStrategy.sol	184	Public function that should be declared external (x7)	Solidity Coding Best Practices	Info	✔ Resolved
	177	Incorrect Solidity version	Solidity Coding Best Practices	Low	✔ Resolved
NotifyHelper.sol	184	Public function that should be declared external (x2)	Solidity Coding Best Practices	Info	✔ Resolved
	198-a	A privileged EOA can call a function that allows to withdraw all funds located in the contract to a needed address	DeFi-Specific	Critical	✔ Resolved
	177	Incorrect Solidity version	Solidity Coding Best Practices	Low	✔ Resolved
	104-b	Unchecked return value (x2)	Control Flow	Medium	✔ Resolved
RewardToken.sol	198-d	A privileged EOA can change protocol parameters	DeFi-Specific	Critical	✔ Resolved
	177	Incorrect Solidity version	Solidity Coding Best Practices	Low	✔ Resolved
	184	Public function that should be declared external	Solidity Coding Best Practices	Info	✔ Resolved
SNXStrategyFullBuyback.sol	177	Incorrect Solidity version	Solidity Coding Best Practices	Low	✔ Resolved
	184	Public function that should be declared external (x5)	Solidity Coding Best Practices	Info	✔ Resolved
SmartVault.sol	177	Incorrect Solidity version	Solidity Coding Best Practices	Low	✔ Resolved

	184	Public function that should be declared external (x11)	Solidity Coding Best Practices	Info	✔ Resolved
WaultStrategyFullBuyback.sol	177	Incorrect Solidity version	Solidity Coding Best Practices	Low	✔ Resolved
	184	Public function that should be declared external (x6)	Solidity Coding Best Practices	Info	✔ Resolved

*Acknowledged – the issue is described to the project team, however the team stated that the function was implemented for any urgent cases to save user's funds.

5. FINDINGS

5.1 Smart Contract Security Analysis

5.1.1 The Bookkeeper.sol contract

Features	*The controller or governance can call the following functions:* **addVaultAndStrategy()** **removeFromVaults()** **removeFromStrategies()** *The controller can call the following functions:* **addVault()** **addStrategy()**

Issues Found

Incorrect Solidity version

Severity: Low

SCW ID: 177

Description: Avoid using complex pragma statements and do not use old solidity versions.

Location: Version used: ['0.7.6', '>=0.4.24<0.8.0', '^0.7.0'].

Recommendations: Using the latest stable pragma compiler.

Status: Fixed.

Public function that should be declared external (x2)

Severity: Info

SCW ID: 184

Description: To save gas, all functions that are not used by other functions of the contract should be declared as external.

Location: lastHardWork(address), lastPpfsChange(address).

Recommendations: The function should be declared as external.

Status: Fixed.

5.1.2 The Controller.sol Contract

Features	*The governor can call the following functions:* **setGovernance()**

setFeeRewardForwarder()

setBookkeeper()

setMintHelper()

setRewardToken()

setNotifyHelper()

setPsVault()

setRewardDistribution()

setPSNumeratorDenominator()

addHardWorker()

removeHardWorker()

addToWhiteList()

removeFromWhiteList()

addVaultsAndStrategies()

addVaultAndStrategy()

doHardWork()

salvage()

salvageStrategy()

Issues Found

A privileged EOA can change protocol parameters (x7)

Severity: Critical

SCW ID: 198-d

Description: Security of user funds handled by the contract fully depends on actions of the third party (the contract owner). Anytime, the owner can change protocol parameters.

Location: setGovernance(), setMintHelper(), setNotifyHelper(), setFeeRewardForwarder(), setPsVault(), setRewardDistribution(), setRewardToken().

Recommendations: We strongly recommend using TimeLock as the governance for such contracts.

Status: Fixed. The Timelock is used as function modifier.

A privileged EOA can change address of token reward distribution

Severity: Low

SCW ID: 198-b

Description: The contract owner can redirect pool/vault rewards.

Location: mintAndDistribute()

Status: Fixed partly. The Timelock is used as function modifier.

A privileged EOA can call a function that allows to withdraw funds to a needed address

Severity: Low

SCW ID: 198-a

Description: We recommend avoiding such functions as the controllerTokenMove(). All the token distribution logic should be decentralized.

Location: controllerTokenMove()

Status: Fixed partly. The Timelock is used as function modifier.

Incorrect Solidity version

Severity: Low

SCW ID: 177

Description: Avoid using complex pragma statements and do not use old solidity versions.

Location: Version used: ['0.7.6', '>=0.4.24<0.8.0', '^0.7.0'].

Recommendations: Using the latest stable pragma compiler.

Status: Fixed.

Comparison to boolean constant

Severity: Info

SCW ID: 170

Description: Boolean constants can be used directly and do not need to be compared to true or false.

Location: addStrategy(address).

Recommendations: Remove the equality to the boolean constant.

Status: Fixed.

Public function that should be declared external

Severity: Info

SCW ID: 184

Description: To save gas, all functions that are not used by other functions of the contract should be declared as external.

Location: addStrategy(address).

Recommendations: The function should be declared as external.

Status: Fixed.

5.1.3 The FeeRewardForwarder.sol Contract

Features	*The governor can call the following functions:* **setConversionPath**

Issues Found

Unchecked Return Value

Severity: Medium

SCW ID: 104-b

Description: approve() function from IERC20 interface returns Boolean value, Ignoring its return value might cause unexpected exceptions

Location: notifyCustomPool()

Recommendations: Check return value of the functions before continuing processing.

Status: Fixed. safeApprove() is used instead of approve().

Incorrect Solidity version

Severity: Low

SCW ID: 177

Description: Avoid using complex pragma statements and do not use old solidity versions.

Location: Version used: ['0.7.6', '>=0.4.24<0.8.0', '^0.7.0'].

Recommendations: Using the latest stable pragma compiler.

Status: Fixed.

Public function that should be declared external

Severity: Info

SCW ID: 184

Description: To save gas, all functions that are not used by other functions of the contract should be declared as external.

Location: setConversionPath(), notifyPsPool(), notifyCustomPool().

Recommendations: The function should be declared as external.

Status: partially fixed.

5.1.4 The MCv2StrategyFullBuyback.sol Contract

Features	The governor can call the following functions: **withdrawAllToVault()** **withdrawToVault()** **investAllUnderlying()** **doHardWork()** **emergencyExit()** **continueInvesting()** **salvage()**

Issues Found

Incorrect Solidity version

Severity: Low

SCW ID: 177

Description: Avoid using complex pragma statements and do not use old solidity versions.

Location: Version used: ['0.7.6', '>=0.4.24<0.8.0', '^0.7.0'].

Recommendations: Using the latest stable pragma compiler.

Status: Fixed.

Public function that should be declared external (x6)

Severity: Info

SCW ID: 184

Description: To save gas, all functions that are not used by other functions of the contract should be declared as external.

Location: rewardTokens(), underlying(), vault(), unsalvageableTokens(address), buyBackRatio(), withdrawToVault(uint256).

Recommendations: The function should be declared as external.

Status: Fixed.

5.1.5 The MintHelper.sol Contract

Features	The governor or controller can call the following functions: **startMinting()** **mint()** **setOperatingFunds()** **changeAdmin()**

Issues Found

A privileged EOA can change protocol parameters

Severity: Critical

SCW ID: 198-d

Description: Security of user funds handled by the contract fully depends on actions of the third party (the contract owner). Anytime, the owner can change protocol parameters.

Location: changeAdmin().

Recommendations: The function should be removed.

Status: Removed.

Incorrect Solidity version

Severity: Low

SCW ID: 177

Description: Avoid using complex pragma statements and do not use old solidity versions.

Location: Version used: '0.7.6', '^0.8.0', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0'

Recommendations: Using the latest stable pragma compiler.

Status: Fixed.

State variables that should be declared constant

Severity: Info

SCW ID: 183

Description: Constant state variables should be declared constant to save gas.

Location: baseRatio, fundsRatio, totalRatio.

Recommendations: Declaring variables as constant.

Status: Fixed.

Public function that should be declared external

Severity: Info

SCW ID: 184

Description: To save gas, all functions that are not used by other functions of the contract should be declared as external.

Location: changeAdmin().

Recommendations: The function should be declared as external.

Status: Deprecated.

5.1.6 The NoopStrategy.sol Contract

Features	*The governor can call the following functions:* **emergencyExit()** **continueInvesting()** **salvage()** **withdrawAllToVault()** **withdrawToVault()** **investAllUnderlying()** **doHardWork()**

Issues Found

Incorrect Solidity version

Severity: Low

SCW ID: 177

Description: Avoid using complex pragma statements and do not use old solidity versions.

Location: Version used: '0.7.6', '>=0.4.24<0.8.0', '^0.7.0'

Recommendations: Using the latest stable pragma compiler.

Status: Fixed.

Public function that should be declared external (x7)

Severity: Info

SCW ID: 184

Description: To save gas, all functions that are not used by other functions of the contract should be declared as external.

Location: rewardTokens(), underlying(), vault(), unsalvageableTokens(address), buyBackRatio(), withdrawToVault(uint256), investAllUnderlying().

Recommendations: The function should be declared as external.

Status: Fixed.

5.1.7 The NotifyHelper.sol Contract

Features	*The governor can call the following functions:* **moveFunds()** **notifyVaults()**

Issues Found

A privileged EOA can call a function that allows to withdraw all funds located in the contract to a needed address

Severity: Critical

SCW ID: 198-a

Description: Security of user funds handled by the contract fully depends on actions of the third party (the contract owner). Anytime, the owner can move the funds to any destination.

Location: moveFunds().

Recommendations: The function should be removed.

Status: Removed.

Unchecked Return Value (x2)

Severity: Medium

SCW ID: 104-b

Description: approve() function from IERC20 interface returns Boolean value, Ignoring its return value might cause unexpected exceptions.

Location: notifyVault(), notifyVaultWithPsToken().

Recommendations: Check return value of the functions before continuing processing.

Status: Fixed. safeApprove() is used instead of approve().

Incorrect Solidity version

Severity: Low

SCW ID: 177

Description: Avoid using complex pragma statements and do not use old solidity versions.

Location: Version used: '0.7.6', '>=0.4.24<0.8.0', '^0.7.0'
Recommendations: Using the latest stable pragma compiler.

Status: Fixed.

Public function that should be declared external (x2)

Severity: Info

SCW ID: 184

Description: To save gas, all functions that are not used by other functions of the contract should be declared as external.

Location: moveFunds(), notifyVaults().

Recommendations: The function should be declared as external.

Status: fixed.

5.1.8 The RewardToken.sol Contract

Features	*The admin can call the following functions:* **changeOwner()** **changeAdmin()** **changeMinter()** **startMinting()**

Issues Found

A privileged EOA can change protocol parameters

Severity: Critical

SCW ID: 198-d

Description: Security of user funds handled by the contract fully depends on actions of the third party (the contract admin). Anytime, the owner can change protocol parameters.

Location: changeMinter().

Recommendations: The minter should be hardcoded or admin should be renounced.

Status: Removed.

Incorrect Solidity version

Severity: Low

SCW ID: 177

Description: Avoid using complex pragma statements and do not use old solidity versions.

Location: Version used: '0.7.6', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0'
Recommendations: Using the latest stable pragma compiler.

Status: Fixed.

Public function that should be declared external

Severity: Info

SCW ID: 184

Description: To save gas, all functions that are not used by other functions of the contract should be declared as external.

Location: startMinting().

Recommendations: The function should be declared as external.

Status: Fixed.

5.1.9 The SNXStrategyFullBuyback.sol Contract

Features	*The governor can call the following functions:* **emergencyExit()** **continueInvesting()** **salvage()** **withdrawAllToVault()** **withdrawToVault()** **investAllUnderlying()** **doHardWork()**

Issues Found

Incorrect Solidity version

Severity: Low

SCW ID: 177

Description: Avoid using complex pragma statements and do not use old solidity versions.

Location: Version used: '0.7.6', '>=0.4.24<0.8.0', '^0.7.0'

Recommendations: Using the latest stable pragma compiler.

Status: Fixed.

Public function that should be declared external (x5)

Severity: Info

SCW ID: 184

Description: To save gas, all functions that are not used by other functions of the contract should be declared as external.

Location: underlying(), vault(), unsalvageableTokens(), buyBackRatio(), withdrawToVault().

Recommendations: The function should be declared as external.

Status: Fixed.

5.1.10 TheSmartVault.sol Contract

Features	*The governor can call the following functions:* **changeActivityStatus()** **doHardWork()** **addRewardToken()**

	removeRewardToken()
	rebalance()
	withdrawAllToVault()
	scheduleUpgrade()
	finalizeUpgrade()
	announceStrategyUpdate()
	finalizeStrategyUpdate()
	setStrategy()

Issues Found

Incorrect Solidity version

Severity: Low

SCW ID: 177

Description: Avoid using complex pragma statements and do not use old solidity versions.

Location: Version used: '0.7.6', '>=0.4.24<0.8.0', '>=0.6.0<0.8.0', '^0.7.0'

Recommendations: Using the latest stable pragma compiler.

Status: Fixed.

Public function that should be declared external (x11)

Severity: Info

SCW ID: 184

Description: To save gas, all functions that are not used by other functions of the contract should be declared as external.

Location: initializeSmartVault(), addRewardToken(), removeRewardToken(), depositFor(), getReward(), rewardTokens(), rewardTokensLength(), notifyTargetRewardAmount(), scheduleUpgrade(), announceStrategyUpdate(), setStrategy()
Recommendations: The function should be declared as external.

Status: Fixed.

5.1.11 The WaultStrategyFullBuyback.sol Contract

Features	*The governor can call the following functions:* **emergencyExit()** **continueInvesting()** **salvage()** **withdrawAllToVault()** **withdrawToVault()** **investAllUnderlying()** **doHardWork()**

Issues Found

Incorrect Solidity version

Severity: Low

SCW ID: 177

Description: Avoid using complex pragma statements and do not use old solidity versions.

Location: Version used: '0.7.6', '>=0.4.24<0.8.0', '^0.7.0'

Recommendations: Using the latest stable pragma compiler.

Status: Fixed.

Public function that should be declared external (x6)

Severity: Info

SCW ID: 184

Description: To save gas, all functions that are not used by other functions of the contract should be declared as external.

Location: rewardTokens(), underlying(), vault(), unsalvageableTokens(), buyBackRatio(), withdrawToVault().

Recommendations: The function should be declared as external.

Status: Fixed.

6. CONCLUSION

The audited contracts are the main part of the Tetu.io ecosystem. Contracts are well written and commented with good readability.

The Tetu team did a lot of work to remove all previously found vulnerabilities, including critical centralization issues. Timelock with announcement logic was provided for every critical contract change.

It's important to point out that the Controller contract still features certain centralization degree: the functions mintAndDistribute() and controllerTokenMove() enable the contract owner to have control over the protocol's token distribution, but with the timelock delay. We are confident that any token distribution should be fully decentralized. However, there are no critical issues left, and users have an ability to monitor all announcements related to the token transfers.

No suspicious functions were revealed during the auditing.

www.defiyield.app