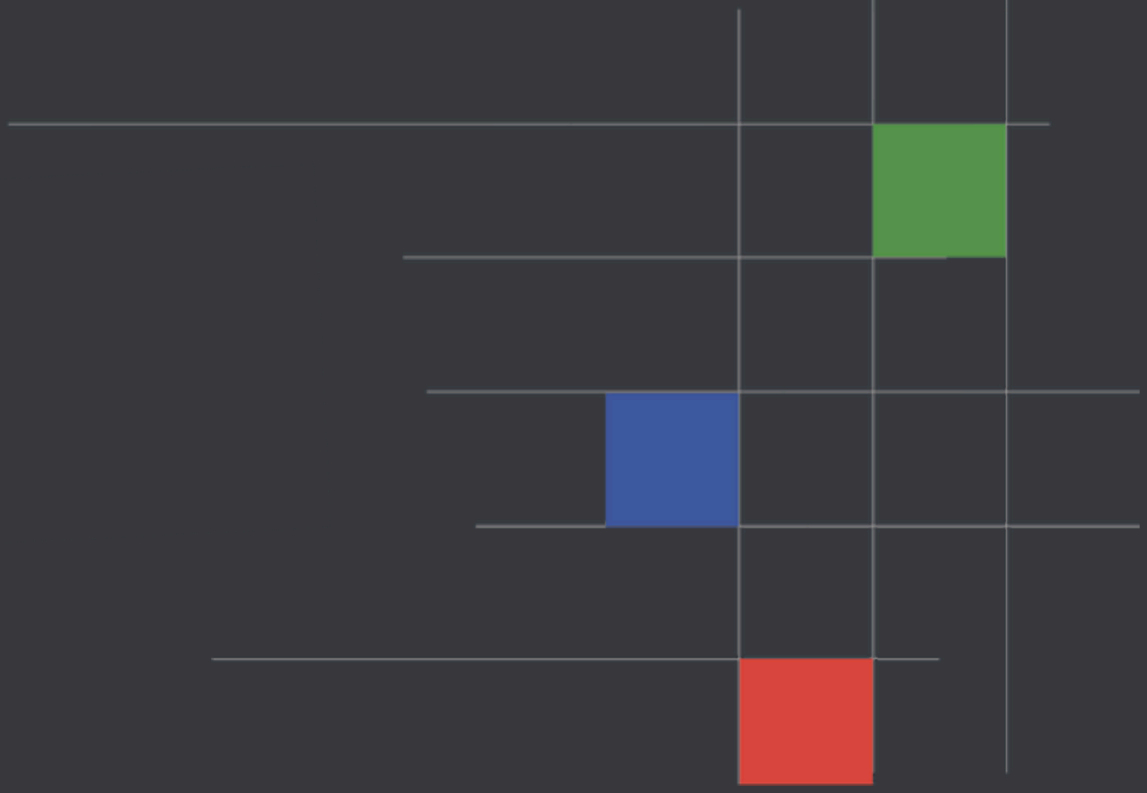
Autofarm

Security Assessment

March 28th, 2021



Disclaimer

CertiK reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product’s IT infrastructure and or source code.

Overview

Project Summary

Project Name	Autofarm
Description	DeFi
Platform	Ethereum; Solidity
Codebase	GitHub Repository
Commit	f32a71e96e080c46c6f0c9a61bd06ad1f643397e 2f17bb88430c5059123fff98675c83978efadeea 3f20703cd1c85943a56ff04448ed0795659f2c03 373389b3a61a790837aac3f10a3cde2b2501546b ec129341efba8bdc70ccb0e113170cb91e472d22

Audit Summary

Delivery Date	Mar. 28th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	Mar. 3rd, 2021 - Mar. 28th, 2021

Vulnerability Summary

Total Issues	44
Total Critical	1
Total Major	7
Total Medium	0
Total Minor	6
Total Informational	30

Executive Summary

This report has been prepared for **Autofarm** smart contract to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

There are a few depending injection contracts in current project:
`_tokenAddress` for contract **TimelockController**;
`AUTO`, `AUTOv2`, `poolInfo[i].want` and `poolInfo[i].strat` for contract **AutoFarmV2**;
`AUTO`, `AUTOv2`, `poolInfo[i].want` and `poolInfo[i].strat` for contract **AutoFarmV2_CrossChain**;
`wantAddress`, `vTokenAddress`, `venusMarkets`, `uniRouterAddress`, `wbnbAddress`, `venusAddress`, `venusDistributionAddress`, `autoFarmAddress`, `AUTOAddress`, `govAddress`, `venusToWantPath` and `earnedToAUTOPath` for contract **StratVLEV2**;
`farmContractAddress`, `wantAddress`, `token0Address`, `token1Address`, `earnedAddress`, `uniRouterAddress`, `wbnbAddress`, `autoFarmAddress`, `AUTOAddress`, `govAddress`, `earnedToAUTOPath`, `earnedToToken0Path`, `earnedToToken1Path`, `token0ToEarnedPath` and `token1ToEarnedPath` for contract **StratX2**;
`farmContractAddress`, `wantAddress`, `token0Address`, `token1Address`, `earnedAddress`, `uniRouterAddress`, `wbnbAddress`, `autoFarmAddress`, `AUTOAddress`, `govAddress`, `earnedToAUTOPath`, `earnedToToken0Path`, `earnedToToken1Path`, `token0ToEarnedPath`, `token1ToEarnedPath`, `MDXAddress`, `MDEXSwapMiningAddress` and `MDXToEarnedPath` for contract **StratX2_MDEX**;
We assume these contracts are valid and non-vulnerable actors, and implementing proper logic to collaborate with current project.

File in Scope

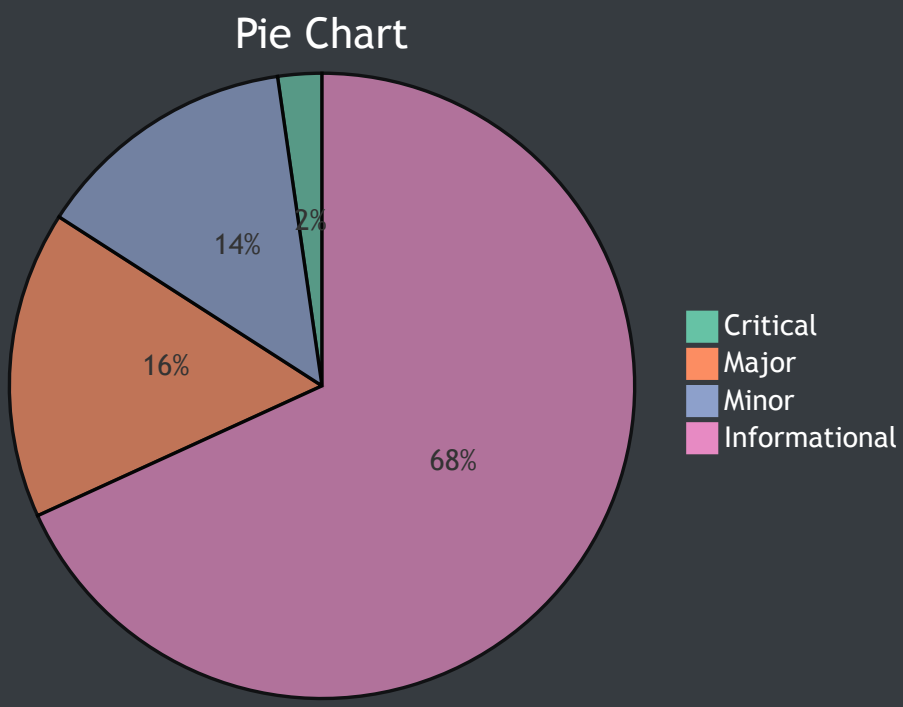
Pre-audit files

ID	Contract	SHA-256 Checksum
AFT	AutoFarmTimelock.sol	de2cc44d7941b3121542c21a8cb6361c2e44526775f31e9d51ee07c2c051e807
AFV	AutoFarmV2.sol	96f38285f7f909cc4c7e0437dc1c2db64c598b58c598de21c664438c512e928b
AFC	AutoFarmV2_CrossChain.sol	7d1eb0aa01d58214c76f9d32114b805bb27172d8a3a556aa6dd2ded91ca15d85
ATV	AUTOv2.sol	33b26cc5b079fd103455a1c6fda8ed56d3832969eb217d5c45b6980745d39145
SVL	StratVLEV2.sol	9bec50e09beb14f4361d436c6641c8cda21cad90da22d19583bb68d9747dda20
STX	StratX.sol	e54ae3b232ed758592ed070a7dfcc92cb76be28a176e493dc5f78ff664f87cb4
SX2	StratX2.sol	d34980ebca3e708eb2d6efc8a55154c9b574fbe05d1d410e826dfaa207fc842a
SXM	StratX2_MDEX.sol	c9179af131a09a71296a45019f68be781e884aebe9d439a97f74be202baefc5a
HAC	helpers/AccessControl.sol	e086c2fff6459a954be48145766653348b2ec0b71d451135001c92c88ff7d039
HCT	helpers/Context.sol	f47f3891d5ca9a606ec4f3355a4d1fd80fe8e3659ffad449747413681b707a71
HER	helpers/ERC20.sol	e78ed4b97cfa0a1b1ca386e857015be8a7998b9c9ddcbbb613e7b98d0918d466
HOW	helpers/Ownable.sol	0cab54a78838ce36779f69349e90d0565fc1d364a32021d3cfc792650fba9125
HPS	helpers/Pausable.sol	a5f199bfa218c27df6333967e3b7bad47350a32383ee42a61089d485f21e5dcd
HRG	helpers/ReentrancyGuard.sol	4f9c6c0aa7a4cf61a782c92d7b6495173f9b8e04c8908aee47a73b766f1cd366
IER	interfaces/IERC20.sol	e71b80ac0695a4021274ed2a2031aff3f85954ea1536b1a1bdb02fa0467cac6d
IP1	interfaces/IPancakeRouter01.sol	7011d5fdf4a12f2ecf51270ce0aa3d0135f116453aa973eb17e2cbda78ee48ee
IP2	interfaces/IPancakeRouter02.sol	c79c0f9cbd1fc22ea7473221ccd716e003ae421a6c0c75a0cc46ef843fb58ed7
IPF	interfaces/IPancakeswapFarm.sol	fa0bb436eb39a9ad8e22ec54afc9ba3af4b1bc679e6b1eecf437f6a350fa9365
LAD	libraries/Address.sol	cb0c20225a5c534ace84ba0806671d53baafb93ad8aea7db4a7e524fc74d5c9a
LES	libraries/EnumerableSet.sol	f1d086945e90e487a85a11164be132bec11d577b2b0b85ecc3ace04aa9873019
LSE	libraries/SafeERC20.sol	b16f06c89e486138168fbe6e46c74c2d9e4669a0c84538cab94336a6e46d4114
LSM	libraries/SafeMath.sol	c064dce5a89ec85e79734e4db348ae28f4fb6a92b97de7fb2fb1c54c70a9312f
LSV	libraries/StratVLEV2.sol	39abb4dd40c6a3a7e9993eaad63cdd2f846721183286d99c8e2ea469b323df03

Post-audit files

ID	Contract	SHA-256 Checksum
AFT	AutoFarmTimelock.sol	3a7026d034394aa0f45756dd575444d0290a29fd61e6759c47027938b9a3992d
AFV	AutoFarmV2.sol	96f38285f7f909cc4c7e0437dc1c2db64c598b58c598de21c664438c512e928b
AFC	AutoFarmV2_CrossChain.sol	7d1eb0aa01d58214c76f9d32114b805bb27172d8a3a556aa6dd2ded91ca15d85
ATV	AUTOv2.sol	33b26cc5b079fd103455a1c6fda8ed56d3832969eb217d5c45b6980745d39145
SVL	StratVLEV2.sol	b790fc0876d7e732caff89e300b265e7fe75ce1441b6fddea534474a0d043cf0
SX2	StratX2.sol	d34980ebca3e708eb2d6efc8a55154c9b574fbe05d1d410e826dfaa207fc842a
SXM	StratX2_MDEX.sol	c9179af131a09a71296a45019f68be781e884aebe9d439a97f74be202baefc5a
HAC	helpers/AccessControl.sol	e086c2fff6459a954be48145766653348b2ec0b71d451135001c92c88ff7d039
HCT	helpers/Context.sol	f47f3891d5ca9a606ec4f3355a4d1fd80fe8e3659ffad449747413681b707a71
HER	helpers/ERC20.sol	e78ed4b97cfa0a1b1ca386e857015be8a7998b9c9ddcbbb613e7b98d0918d466
HOW	helpers/Ownable.sol	0cab54a78838ce36779f69349e90d0565fc1d364a32021d3cfc792650fba9125
HPS	helpers/Pausable.sol	a5f199bfa218c27df6333967e3b7bad47350a32383ee42a61089d485f21e5dcd
HRG	helpers/ReentrancyGuard.sol	4f9c6c0aa7a4cf61a782c92d7b6495173f9b8e04c8908aee47a73b766f1cd366
IER	interfaces/IERC20.sol	e71b80ac0695a4021274ed2a2031aff3f85954ea1536b1a1bdb02fa0467cac6d
IP1	interfaces/IPancakeRouter01.sol	7011d5fdf4a12f2ecf51270ce0aa3d0135f116453aa973eb17e2cbda78ee48ee
IP2	interfaces/IPancakeRouter02.sol	c79c0f9cbd1fc22ea7473221ccd716e003ae421a6c0c75a0cc46ef843fb58ed7
IPF	interfaces/IPancakeswapFarm.sol	fa0bb436eb39a9ad8e22ec54afc9ba3af4b1bc679e6b1eecf437f6a350fa9365
LAD	libraries/Address.sol	cb0c20225a5c534ace84ba0806671d53baafb93ad8aea7db4a7e524fc74d5c9a
LES	libraries/EnumerableSet.sol	f1d086945e90e487a85a11164be132bec11d577b2b0b85ecc3ace04aa9873019
LSE	libraries/SafeERC20.sol	b16f06c89e486138168fbe6e46c74c2d9e4669a0c84538cab94336a6e46d4114
LSM	libraries/SafeMath.sol	c064dce5a89ec85e79734e4db348ae28f4fb6a92b97de7fb2fb1c54c70a9312f

Findings



ID	Title	Type	Severity	Resolved
AFT-01	Functions Should Be Declared External	Optimization	Informational	!
AFT-02	Missing Checks for Reentrancy	Logic Issue	Major	✓
AFT-03	Missing Events for Significant Transactions	Optimization	Informational	!
AFT-04	Missing Return Value Handling	Logic Issue	Minor	✓
AFT-05	Bad Naming Convention	Coding Style	Informational	!
AFV-01	Variables Should Be Declared Constant	Optimization	Informational	!

AFV-02	Functions Should Be Declared External	Optimization	Informational	
AFV-03	Missing Checks for Reentrancy	Logic Issue	Major	
AFV-04	Bad Naming Convention	Coding Style	Informational	
AFV-05	Missing Checks for Human Errors	Logic Issue	Informational	
AFV-06	Missing Return Value Handling	Logic Issue	Minor	
AFV-07	Centralization Risks	Logic Issue	Major	
AFC-01	Redundant Variables, Data Structures and Functions	Optimization	Informational	
AFC-02	Functions Should Be Declared External	Optimization	Informational	
AFC-03	Bad Naming Convention	Coding Style	Informational	
AFC-04	Missing Checks for Human Errors	Logic Issue	Informational	
AFC-05	Missing Return Value Handling	Logic Issue	Minor	
AFC-06	Centralization Risks	Logic Issue	Major	
SVL-01	Missing Return Value Handling	Logic Issue	Minor	
SVL-02	No Entrance Fees When Vault Is Empty	Logic Issue	Informational	
SVL-03	Redundant Type Conversion	Coding Style	Informational	
SVL-04	Functions Should Be Declared External	Optimization	Informational	
SVL-05	Missing Checks for Reentrancy	Logic Issue	Major	
SVL-06	Lack of Access Control	Logic Issue	Major	

SVL-07	Missing Events for Significant Transactions	Optimization	Informational	
SVL-08	Unable to Withdraw Earned Token When Paused	Logic Issue	Informational	
SVL-09	Missing Checks for Input	Optimization	Informational	
SVL-10	Imprecise Comparison	Optimization	Informational	
SVL-11	Integer Overflow	Mathematical Operations	Informational	
SVL-12	Conditions Could Be Merged	Optimization	Informational	
SVL-13	Keep the Code DRY	Coding Style	Informational	
SX2-01	Functions Should Be Declared External	Optimization	Informational	
SX2-02	Missing Checks for Reentrancy	Logic Issue	Major	
SX2-03	No Entrance Fees When Vault Is Empty	Logic Issue	Informational	
SX2-04	Missing Access Control	Logic Issue	Critical	
SX2-05	Unable to Withdraw Earned Token When Paused	Logic Issue	Informational	
SX2-06	Missing Return Value Handling	Logic Issue	Informational	
SX2-07	Conditions Could Be Merged	Optimization	Informational	
SX2-08	Short Waiting Period for Transactions	Logic Issue	Minor	
SX2-09	Integer Overflow	Mathematical Operations	Informational	
SX2-10	Imprecise Comparison	Optimization	Informational	
SX2-11	Keep the Code DRY	Coding Style	Informational	

SX2-12	Centralization Risks	Logic Issue	Minor	
LAD-01	Corner Case for Non-contract Caller Check	Volatile Code	Informational	

AFT-01: Functions Should Be Declared External

Type	Severity	Location
Optimization	Informational	AutoFarmTimelock.sol

Description:

Functions which are never called internally within the contract should have external visibility. For example, `getTimestamp`, `getMinDelay`, `schedule`, `scheduleBatch`, `cancel`, `execute`, `executeBatch`, `setDevWalletAddress`, `scheduleSet`, `executeSet`, `withdrawBNB`, `withdrawBEP20`, `add`, `earn`, `farm`, `pause`, `unpause`, `rebalance`, `deleverageOnce`, `wrapBNB`, `noTimeLockFunc1`, `noTimeLockFunc2` and `noTimeLockFunc3`.

Recommendation:

We recommend changing the visibility of the aforementioned functions to `external`.

Alleviation:

No alleviation.

AFT-02: Missing Checks for Reentrancy

Type	Severity	Location
Logic Issue	Major	AutoFarmTimelock.sol: L374, L396, L447, L538

Description:

Function `execute`, `executeBatch`, `_call` and `executeSet` have state updates or event emits after external calls and thus are vulnerable to reentrancy attack.

Recommendation:

We recommend applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation:

The development team heeded our advice and resolved this issue. The fixing is reflected in commit 99c9c74ce1f863ac3332032ae419e6b4d0dbc9d1.

AFT-03: Missing Events for Significant Transactions

Type	Severity	Location
Optimization	Informational	AutoFarmTimelock.sol: L489

Description:

Function `setDevWalletAddress` updates `devWalletAddress` which is a key role of contract `TimelockController`. Missing event makes it difficult to track off-chain role changes. An event should be emitted for significant transactions like this.

Recommendation:

We recommend emitting an event to log the update of `devWalletAddress` in `setDevWalletAddress`.

Alleviation:

No alleviation.

AFT-04: Missing Return Value Handling

Type	Severity	Location
Logic Issue	🔵 Minor	AutoFarmTimelock.sol: L575

Description:

`transfer` is not a void-returning function per IERC20 interface. Ignoring the return value might cause some unexpected exception, especially if the callee function doesn't revert automatically when failing.

Recommendation:

We recommend checking the output of `transfer` before continuing processing.

Alleviation:

The development team heeded our advice and resolved this issue. The fixing is reflected in commit 9fec6f472967bf1158951d7144d662247d2ffa68.

AFT-05: Bad Naming Convention

Type	Severity	Location
Coding Style	Informational	AutoFarmTimelock.sol: L578

Description:

Function name `add` collides with `SafeMath.add`. Solidity suggests a naming convention that should be followed.

Recommendation:

We recommend modifying the function name `add` to `addStrat`.

Alleviation:

No alleviation.

AFV-01: Variables Should Be Declared Constant

Type	Severity	Location
Optimization	Informational	AutoFarmV2.sol

Description:

Some variables are not modified within contracts and thus could be declared constant. For example, `AUTO`, `AUTOv2`, `burnAddress`, `ownerAUTOReward`, `AUTOMaxSupply`, `AUTOPerBlock` and `startBlock`.

Recommendation:

We recommend declaring the aforementioned variables as `constant`.

Alleviation:

No alleviation.

AFV-02: Functions Should Be Declared External

Type	Severity	Location
Optimization	Informational	AutoFarmV2.sol

Description:

Functions which are never called internally within the contract should have external visibility. For example, `add`, `set`, `deposit`, `withdrawAll`, `emergencyWithdraw` and `inCaseTokensGetStuck`.

Recommendation:

We recommend changing the visibility of the aforementioned functions to `external`.

Alleviation:

No alleviation.

AFV-03: Missing Checks for Reentrancy

Type	Severity	Location
Logic Issue	Major	AutoFarmV2.sol: L110, L134, L211

Description:

Function `add`, `set` and `updatePool` have state updates or event emits after external calls and thus are vulnerable to reentrancy attack.

Recommendation:

We recommend applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation:

(**Autofarm Team - Response**)
Access to add and set restricted to only admin. UpdatePool is safe to be called multiple times in same transaction, as in massUpdatePools.

AFV-04: Bad Naming Convention

Type	Severity	Location
Coding Style	Informational	AutoFarmV2.sol: L110

Description:

Function name `add` collides with `SafeMath.add`. Solidity suggests a naming convention that should be followed.

Recommendation:

We recommend modifying the function name `add` to `addPool`.

Alleviation:

No alleviation.

AFV-05: Missing Checks for Human Errors

Type	Severity	Location
Logic Issue	Informational	AutoFarmV2.sol: L110

Description:

Considering adding duplicated pools by calling `add` might cause errors in future transactions, checking if the input pool/strategy already exists would be helpful to avoid human errors.

Recommendation:

We recommend adding a check to guarantee the input pool/strategy does not exist in `poolInfo`.

Alleviation:

No alleviation.

AFV-06: Missing Return Value Handling

Type	Severity	Location
Logic Issue	Minor	AutoFarmV2.sol: L335, L347, L349

Description:

`IStrategy.withdraw` and `IERC20.transfer` are not void-returning functions. Ignoring the return value might cause some unexpected exception, especially if the callee function doesn't revert automatically when failing.

Recommendation:

We recommend checking the output of the aforementioned functions before continuing processing.

Alleviation:

(**Autofarm Team - Response**)
Functions revert upon failure.
We only do `.transfer` for `AUTOv2`, which is a standard ERC20 contract.

AFV-07: Centralization Risks

Type	Severity	Location
Logic Issue	Major	AutoFarmV2.sol: L353

Description:

Function `inCaseTokensGetStuck` at the aforementioned line allows the owner to drain tokens except `AUTOv2` from the contract.

Recommendation:

We recommend the team to review the design and ensure minimum centralization risk. One of our recommendations is to remove the function `inCaseTokensGetStuck`.

Alleviation:

(**Autofarm Team - Response**)
Contract is designed to never hold any tokens other than AUTOv2. All want tokens get sent into strategy contracts (ie StratX2, StratVLEV2 etc).

AFC-01: Redundant Variables, Data Structures and Functions

Type	Severity	Location
Optimization	Informational	AutoFarmV2_CrossChain.sol

Description:

There is no reward calculation based on accumulated AUTO per share within the contract `AutoFarmV2_CrossChain`, so all variables, data structures and functions related to it could be removed.

Recommendation:

We recommend removing all variables, data structures and functions for reward calculation based on accumulated AUTO per share.

Alleviation:

The development team has heeded our advice and resolved this issue. The fixing is reflected in commit 4837b9341f834e5c11d402b422eaa7ef753b8827

AFC-02: Functions Should Be Declared External

Type	Severity	Location
Optimization	Informational	AutoFarmV2_CrossChain.sol

Description:

Functions which are never called internally within the contract should have external visibility. For example, `add`, `set`, `deposit`, `withdrawAll`, `emergencyWithdraw` and `inCaseTokensGetStuck`.

Recommendation:

We recommend changing the visibility of the aforementioned functions to `external`.

Alleviation:

No alleviation.

AFC-03: Bad Naming Convention

Type	Severity	Location
Coding Style	Informational	AutoFarmV2_CrossChain.sol: L81

Description:

Function name `add` collides with `SafeMath.add` . Solidity suggests a naming convention that should be followed.

Recommendation:

We recommend modifying the function name `add` to `addPool` .

Alleviation:

No alleviation.

AFC-04: Missing Checks for Human Errors

Type	Severity	Location
Logic Issue	Informational	AutoFarmV2_CrossChain.sol: L81

Description:

Considering adding duplicated pools by calling `add` might cause errors in future transactions, checking if the input pool/strategy already exists would be helpful to avoid human errors.

Recommendation:

We recommend adding a check to guarantee the input pool/strategy does not exist in `poolInfo`.

Alleviation:

No alleviation.

AFC-05: Missing Return Value Handling

Type	Severity	Location
Logic Issue	Minor	AutoFarmV2_CrossChain.sol: L231

Description:

`IStrategy.withdraw` is not a void-returning. Ignoring the return value might cause some unexpected exception, especially if the callee function doesn't revert automatically when failing.

Recommendation:

We recommend checking the output of function `IStrategy.withdraw` before continuing processing.

Alleviation:

(**Autofarm Team - Response**)
Functions revert upon failure.

AFC-06: Centralization Risks

Type	Severity	Location
Logic Issue	Major	AutoFarmV2_CrossChain.sol: L238

Description:

Function `inCaseTokensGetStuck` at the aforementioned line allows the owner to drain tokens from the contract.

Recommendation:

We recommend the team to review the design and ensure minimum centralization risk. One of our recommendations is to remove the function `inCaseTokensGetStuck` .

Alleviation:

(**Autofarm Team - Response**)
All want tokens get sent into strategy contracts (ie StratX2, StratVLEV2 etc).

SVL-01: Missing Return Value Handling

Type	Severity	Location
Logic Issue	Minor	StratVLEV2.sol: L214, L219, L223, L230, L414, L448, L469

Description:

Functions called at the aforementioned lines are not void-returning functions. Ignoring the return value might cause some unexpected exception, especially if the callee function doesn't revert automatically when failing.

Recommendation:

We recommend checking the output of functions at the aforementioned lines before continuing processing.

Alleviation:

(**Autofarm Team - Response**)
Functions revert upon failure.

SVL-02: No Entrance Fees When Vault Is Empty

Type	Severity	Location
Logic Issue	Informational	StratVLEV2.sol: L244

Description:

`entranceFeeFactor/entranceFeeFactorMax` will not be applied to `_wantAmt` when `wantLockedTotal() == 0` or `sharesTotal == 0`. Is this an intended design?

Alleviation:

(**Autofarm Team - Response**)
Checked, and yes, is intended.

SVL-03: Redundant Type Conversion

Type	Severity	Location
Coding Style	Informational	StratVLEV2.sol: L255

Description:

`address(msg.sender)` is equivalent to `msg.sender`. The type conversion is not necessary.

Recommendation:

We recommend changing `address(msg.sender)` to `msg.sender`.

Alleviation:

No alleviation.

SVL-04: Functions Should Be Declared External

Type	Severity	Location
Optimization	Informational	StratVLEV2.sol

Description:

Functions which are never called internally within the contract should have external visibility. For example, `deposit`, `farm`, `deleverageOnce`, `pause`, `resetAllowances`, `setEntranceFeeFactor`, `setWithdrawFeeFactor`, `setControllerFee`, `setbuyBackRate`, `setGov`, `setOnlyGov`, `setUniRouterAddress`, `setDeleverAmtFactorMax`, `setDeleverAmtFactorSafe` and `inCaseTokensGetStuck`.

Recommendation:

We recommend changing the visibility of the aforementioned functions to `external`.

Alleviation:

No alleviation.

SVL-05: Missing Checks for Reentrancy

Type	Severity	Location
Logic Issue	Major	StratVLEV2.sol: L269, L310, L365, L435, L580

Description:

Function `_farm`, `_deleverageOnce`, `_deleverage`, `earn` and `updateBalance` have state updates or event emits after external calls and thus are vulnerable to reentrancy attack.

Recommendation:

We recommend applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation:

The development team heeded our advice and resolved this issue. The fixing is reflected in commit 99c9c74ce1f863ac3332032ae419e6b4d0dbc9d1.

SVL-06: Lack of Access Control

Type	Severity	Location
Logic Issue	Major	StratVLEV2.sol: L303

Description:

If `onlyGov` is set to `false`, there will be no access control for `deleverageOnce`, which means anyone would change the strategy by calling `deleverageOnce`. Is this an intended design?

Alleviation:

The development team heeded our advice and resolved this issue. The fixing is reflected in commit 99c9c74ce1f863ac3332032ae419e6b4d0dbc9d1.

SVL-07: Missing Events for Significant Transactions

Type	Severity	Location
Optimization	Informational	StratVLEV2.sol: L422

Description:

Function `rebalance` updates `borrowRate` and `borrowDepth`. An event (`StratRebalance`, which is already implemented at line 169) should be emitted for significant transactions like this.

Recommendation:

We recommend emitting an event to log the update of `borrowRate` and `borrowDepth` in `rebalance`.

Alleviation:

No alleviation.

SVL-08: Unable to Withdraw Earned Token When Paused

Type	Severity	Location
Logic Issue	Informational	StratVLEV2.sol: L435

Description:

Function `earn` is restricted to `whenNotPaused`. It means when the contract is paused, users are allowed to withdraw token already in this contract's account, but they are not allowed to withdraw pending earned token. Is this an intended design?

Alleviation:

(**Autofarm Team - Response**)
Checked, and yes, is intended.

SVL-09: Missing Checks for Input

Type	Severity	Location
Optimization	Informational	StratVLEV2.sol: L462, L497

Description:

For some functions, checking the input could simplify calculations and thus save gas. For example, checking if `_earnedAmt == 0` for `buyBack` and `_wantAmt == 0` for `withdraw`.

Recommendation:

We recommend checking input values for `buyBack` and `withdraw` to simplify operations.

Alleviation:

No alleviation.

SVL-10: Imprecise Comparison

Type	Severity	Location
Optimization	Informational	StratVLEV2.sol: L463

Description:

As a `uint256` type variable `buyBackRate` will never be negative.

Recommendation:

We recommend changing the condition `buyBackRate <= 0` to `buyBackRate == 0`

Alleviation:

No alleviation.

SVL-11: Integer Overflow

Type	Severity	Location
Mathematical Operations	Informational	StratVLEV2.sol: L474

Description:

Although integer overflows would not happen if some variables such as `now` are within regular ranges, `SafeMath` is still highly recommended for mathematical operations, if gas costs are not considered as a most significant factor in implementations, to prevent exceptions.

Recommendation:

We recommend applying `SafeMath.add` at the aforementioned line:

```
        now.add(600)
```

Alleviation:

The development team heeded our advice and resolved this issue. The fixing is reflected in commit 99c9c74ce1f863ac3332032ae419e6b4d0dbc9d1.

SVL-12: Conditions Could Be Merged

Type	Severity	Location
Optimization	Informational	StratVLEV2.sol: L485-486

Description:

Code at the aforementioned lines

```
        if (_earnedAmt > 0) {
            if (controllerFee > 0) {
                ...
            }
        }
```

could be simplified to

```
        if (_earnedAmt > 0 && controllerFee > 0) {
            ...
        }
```

Alleviation:

No alleviation.

Recommendation:

We recommend grouping the conditions at the aforementioned lines.

SVL-13: Keep the Code DRY

Type	Severity	Location
Coding Style	Informational	StratVLEV2.sol

Description:

The following check

```
        require(msg.sender == govAddress, "!gov");
```

is performed multiple times within contract `StratVLEV2`. Extracting the logic and setting a modifier to check if `msg.sender == govAddress` could contribute to code optimization.

Recommendation:

We recommend implementing a modifier to check if `msg.sender == govAddress` and applying the modifier when the access role is restricted to `govAddress`.

Alleviation:

The development team heeded our advice and resolved this issue. The fixing is reflected in commit 99c9c74ce1f863ac3332032ae419e6b4d0dbc9d1.

SX2-01: Functions Should Be Declared External

Type	Severity	Location
Optimization	Informational	StratX2.sol

Description:

Functions which are never called internally within the contract should have external visibility. For example, `deposit`, `farm`, `earn`, `convertDustToEarned`, `pause`, `setEntranceFeeFactor`, `setWithdrawFeeFactor`, `setControllerFee`, `setbuyBackRate`, `setGov`, `setOnlyGov`, `setUniRouterAddress`, `inCaseTokensGetStuck` and `wrapBNB`.

Recommendation:

We recommend changing the visibility of the aforementioned functions to `external`.

Alleviation:

No alleviation.

SX2-02: Missing Checks for Reentrancy

Type	Severity	Location
Logic Issue	Major	StratX.sol: L133, L224

Description:

Function `deposit` and `earn` have state updates or event emits after external calls and thus are vulnerable to reentrancy attack.

Recommendation:

We recommend applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation:

The development team heeded our advice and resolved this issue. The fixing is reflected in commit 2f17bb88430c5059123fff98675c83978efadeea.

SX2-03: No Entrance Fees When Vault Is Empty

Type	Severity	Location
Logic Issue	Informational	StratX2.sol: L146

Description:

`entranceFeeFactor/entranceFeeFactorMax` will not be applied to `_wantAmt` when `wantLockedTotal == 0` or `sharesTotal == 0` . Is this an intended design?

Alleviation:

(**Autofarm Team - Response**)
Checked, and yes, is intended.

SX2-04: Missing Access Control

Type	Severity	Location
Logic Issue	Critical	StratX2.sol: L164

Description:

Function `farm` is `public` and thus can be called by anyone. When `!isAutoComp` and `farm` is called, `wantAddress` token will be sent to `farmContractAddress` . Because function `withdraw` withdraws token from `farmContractAddress` only when `isAutoComp` , users will not be able to withdraw token sent to `farmContractAddress` when `!isAutoComp` .

Recommendation:

We recommend checking if `isAutoComp` is true in function `farm` .

Alleviation:

The development team heeded our advice and resolved this issue. The fixing is reflected in commit 2f17bb88430c5059123fff98675c83978efadeea.

SX2-05: Unable to Withdraw Earned Token When Paused

Type	Severity	Location
Logic Issue	Informational	StratX2.sol: L224

Description:

Function `earn` is restricted to `whenNotPaused`. It means when the contract is paused, users are allowed to withdraw token already in this contract's account, but they are not allowed to withdraw pending earned token. Is this an intended design?

Alleviation:

(**Autofarm Team - Response**)
Checked, and yes, is intended.

SX2-06: Missing Return Value Handling

Type	Severity	Location
Logic Issue	🔵 Informational	StratX2.sol: L295

Description:

`IPancakeRouter02.addLiquidity` is not a void-returning. Ignoring the return value might cause some unexpected exception, especially if the callee function doesn't revert automatically when failing.

Recommendation:

We recommend checking the output of function `IPancakeRouter02.addLiquidity` before continuing processing.

Alleviation:

No alleviation.

SX2-07: Conditions Could Be Merged

Type	Severity	Location
Optimization	Informational	StratX2.sol: L341-343

Description:

Code at the aforementioned lines:

```
        if (_earnedAmt > 0) {
            // Performance fee
            if (controllerFee > 0) {
                ...
            }
        }
```

could be simplified to

```
        if (_earnedAmt > 0 && controllerFee > 0) {
            ...
        }
```

Recommendation:

We recommend grouping the conditions at the aforementioned lines.

Alleviation:

No alleviation.

SX2-08: Short Waiting Period for Transactions

Type	Severity	Location
Logic Issue	Minor	StratX2.sol: L267, L279, L303, L333, L375, L394

Description:

The deadlines of reverting transactions are set to `now+60` at the aforementioned lines. 60 seconds is short, especially when the network is under heavy load, for a waiting period.

Recommendation:

We recommend setting a longer waiting period such as 600.

Alleviation:

The development team heeded our advice and resolved this issue. The fixing is reflected in commit 2f17bb88430c5059123fff98675c83978efadeea.

SX2-09: Integer Overflow

Type	Severity	Location
Mathematical Operations	Informational	StratX2.sol: L267, L279, L303, L333, L375, L394

Description:

Although integer overflows would not happen if some variables such as `now` are within regular ranges, `SafeMath` is still highly recommended for mathematical operations, if gas costs are not considered as a most significant factor in implementations, to prevent exceptions.

Recommendation:

We recommend applying `SafeMath.add` at the aforementioned line.

Alleviation:

The development team heeded our advice and resolved this issue. The fixing is reflected in commit 2f17bb88430c5059123fff98675c83978efadeea.

SX2-10: Imprecise Comparison

Type	Severity	Location
Optimization	Informational	StratX2.sol: L313

Description:

As a `uint256` type variable `buyBackRate` will never be negative.

Recommendation:

We recommend changing the condition `buyBackRate <= 0` to `buyBackRate == 0`

Alleviation:

No alleviation.

SX2-11: Keep the Code DRY

Type	Severity	Location
Coding Style	Informational	StratX2.sol

Description:

The following check

```
        require(msg.sender == govAddress, "!gov");
```

is performed multiple times within contract `StratX2` . Extracting the logic and setting a modifier to check if `msg.sender == govAddress` could contribute to code optimization.

Recommendation:

We recommend implementing a modifier to check if `msg.sender == govAddress` and applying the modifier when the access role is restricted to `govAddress` .

Alleviation:

The development team heeded our advice and resolved this issue. The fixing is reflected in commit 2f17bb88430c5059123fff98675c83978efadeea.

SX2-12: Centralization Risks

Type	Severity	Location
Logic Issue	🔵 Minor	StratX2.sol: L450

Description:

Function `inCaseTokensGetStuck` at the aforementioned line allows the owner to drain tokens other than `earnedAddress` and `wantAddress` from the contract.

Recommendation:

We recommend the team to review the design and ensure minimum centralization risk. One of our recommendations is to remove the function `inCaseTokensGetStuck`.

Alleviation:

(**Autofarm Team - Response**)
Contract is designed to never hold any tokens other than earnedAddress, wantAddress, token0 and token1. The later 2 are allowed to be withdrawn to cater for inefficiencies in strategy, causing dust to accumulate, and further catering for contingency in case convertDustToEarned fails.

LAD-01: Corner Case for Non-contract Caller Check

Type	Severity	Location
Volatile Code	Informational	libraries/Address.sol: L22

Description:

`Address.isContract` cannot 100% guarantee the caller is a non-contract user. For example, EXTCODESIZE returns 0 if it is called from the constructor of another contract. Please consider if this is a problem for the project.

Recommendation:

We recommend checking `msg.sender == tx.origin` to exclude contract calls.

Alleviation:

No alleviation.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a `struct` assignment operation affecting an in-memory `struct` rather than an instorage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete` .

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a `constructor` assignment imposing different `require` statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as `constant` contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.

Icons explanation

 : Issue resolved

 : Issue not resolved / Acknowledged. The team will be fixing the issues in the own timeframe.

 : Issue partially resolved. Not all instances of an issue was resolved.