CURVE FINANCE STABLESWAPNG SECURITY AUDIT REPORT

November 1, 2023

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of the Client. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals

- Build an independent view of the project's architecture.
- Identifying logical flaws.

2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the clients' codes.
- Code check with the use of static analyzers (i.e Slither, Mythril, etc).

Stage goal

Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- Exploits PoC development with the use of such programs as Brownie and Hardhat.

Stage goal

Detect inconsistencies with the desired model.

4. Consolidation of the auditors' interim reports into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals

- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Client with an interim report.

5. Bug fixing & re-audit:

- The Client either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Client with a re-audited report.

6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Conduct the final check of the code deployed on the mainnet.
- Provide the Customer with a public audit report.

Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

1.3 Project Overview

Curve StableSwapNG provides a mechanism to create cross-markets for stablecoins and pegged assets (e.g. stETH / ETH) in a way which could be called "Uniswap with leverage". It is a fully autonomous market-maker for stablecoins and pegged assets with the minimal price slippage, as well as an efficient "fiat savings account" for liquidity providers on the other side.

1.4 Project Dashboard

Project Summary

Title	Description
Client	Curve Finance
Project name	StableSwapNG
Timeline	September 06 2023 - October 26 2023
Number of Auditors	3

Project Log

Date	Commit Hash	Note
07.09.2023	8c78731ed43c22e6bcdcb5d39b0a7d02f8cb0386	Commit for the audit
10.10.2023	bff1522b30819b7b240af17ccfb72b0effbf6c47	Commit for the re-audit
13.10.2023	b5a073c0a8eb1e6281a23d029b7995c2dec261ac	Commit with the ERC4626 logic
26.10.2023	d564a9f43ef33062b2de3ee95a710fc167067aa9	Commit for deploy

Project Scope

The audit covered the following files:

File name	Link
CurveStableSwapFactoryNG.vy	CurveStableSwapFactoryNG.vy
CurveStableSwapMetaNG.vy	CurveStableSwapMetaNG.vy
CurveStableSwapNGMath.vy	CurveStableSwapNGMath.vy
CurveStableSwapNG.vy	CurveStableSwapNG.vy
LiquidityGauge.vy	LiquidityGauge.vy

Deployments

Ethereum:mainnet

File name	Contract deployed on mainnet	Comment
CurveStableSwapNGMath.vy	0x20D1c021525C85D9617Ccc64D8f547d5f730118A	
CurveStableSwapNGViews.vy	0x87DD13Dd25a1DBde0E1EdcF5B8Fa6cfff7eABCaD	
CurveStableSwapFactoryNG.vy	0x6A8cbed756804B16E05E741eDaBd5cB544AE21bf	
CurveStableSwapNG.vy	0x3E3B5F27bbf5CC967E074b70E9f4046e31663181	
CurveStableSwapMetaNG.vy	0x19bd1AB34d6ABB584b9C1D5519093bfAA7f6c7d2	
LiquidityGauge.vy	0xF5617D4f7514bE35fce829a1C19AE7f6c9106979	

Arbitrum:mainnet

File name	Contract deployed on mainnet	Comment

File name	Contract deployed on mainnet	Comment
CurveStableSwapNGMath.vy	0x3d6cB2F6DcF47CDd9C13E4e3beAe9af041d8796a	evm-version paris
CurveStableSwapNGViews.vy	0xC1b393EfEF38140662b91441C6710Aa704973228	evm-version paris
CurveStableSwapFactoryNG.vy	0x9AF14D26075f142eb3F292D5065EB3faa646167b	evm-version paris
CurveStableSwapNG.vy	0x76303e4fDcA0AbF28aB3ee42Ce086E6503431F1D	evm-version paris
CurveStableSwapMetaNGP.vy	0xd125E7a0cEddF89c6473412d85835450897be6Dc	evm-version paris

Optimism:mainnet

File name	Contract deployed on mainnet	Comment
CurveStableSwapNGMath.vy	0x8b3EFBEfa6eD222077455d6f0DCdA3bF4f3F57A6	evm-version paris
CurveStableSwapNGViews.vy	0x506F594ceb4E33F5161139bAe3Ee911014df9f7f	evm-version paris
CurveStableSwapFactoryNG.vy	0x5eeE3091f747E60a045a2E715a4c71e600e31F6E	evm-version paris
CurveStableSwapNG.vy	0x87FE17697D0f14A222e8bEf386a0860eCffDD617	evm-version paris
CurveStableSwapMetaNG.vy	0x1764ee18e8B3ccA4787249Ceb249356192594585	evm-version paris

Base:mainnet

File name	Contract deployed on mainnet	Comment
CurveStableSwapNGMath.vy	0x506F594ceb4E33F5161139bAe3Ee911014df9f7f	evm-version paris
CurveStableSwapNGViews.vy	0x87FE17697D0f14A222e8bEf386a0860eCffDD617	evm-version paris
CurveStableSwapFactoryNG.vy	0xd2002373543Ce3527023C75e7518C274A51ce712	evm-version paris
CurveStableSwapNG.vy	0x1764ee18e8B3ccA4787249Ceb249356192594585	evm-version paris
CurveStableSwapMetaNG.vy	0x5eee3091f747e60a045a2e715a4c71e600e31f6e	evm-version paris

Linea:mainnet

File name	Contract deployed on mainnet	Comment
CurveStableSwapNGMath.vy	0x8b3EFBEfa6eD222077455d6f0DCdA3bF4f3F57A6	evm-version paris
CurveStableSwapNGViews.vy	0x506F594ceb4E33F5161139bAe3Ee911014df9f7f	evm-version paris
CurveStableSwapFactoryNG.vy	0x5eeE3091f747E60a045a2E715a4c71e600e31F6E	evm-version paris
CurveStableSwapNG.vy	0x87FE17697D0f14A222e8bEf386a0860eCffDD617	evm-version paris
CurveStableSwapMetaNG.vy	0x1764ee18e8b3cca4787249ceb249356192594585	evm-version paris

Scroll:mainnet

File name	Contract deployed on mainnet	Comment
CurveStableSwapNGMath.vy	0x8b3EFBEfa6eD222077455d6f0DCdA3bF4f3F57A6	evm-version paris
CurveStableSwapNGViews.vy	0x506F594ceb4E33F5161139bAe3Ee911014df9f7f	evm-version paris
CurveStableSwapFactoryNG.vy	0x5eeE3091f747E60a045a2E715a4c71e600e31F6E	evm-version paris
CurveStableSwapNG.vy	0x87FE17697D0f14A222e8bEf386a0860eCffDD617	evm-version paris
CurveStableSwapMetaNG.vy	0x1764ee18e8B3ccA4787249Ceb249356192594585	evm-version paris

Polygon zkevm:mainnet

File name	Contract deployed on mainnet	Comment
CurveStableSwapNGMath.vy	0x506F594ceb4E33F5161139bAe3Ee911014df9f7f	evm-version paris
CurveStableSwapNGViews.vy	0x87fe17697d0f14a222e8bef386a0860ecffdd617	evm-version paris
CurveStableSwapFactoryNG.vy	0xd2002373543Ce3527023C75e7518C274A51ce712	evm-version paris
CurveStableSwapNG.vy	0x1764ee18e8b3cca4787249ceb249356192594585	evm-version paris
CurveStableSwapMetaNG.vy	0x5eee3091f747e60a045a2e715a4c71e600e31f6e	evm-version paris

Gnosis:mainnet

File name	Contract deployed on mainnet	Comment
CurveStableSwapNGMath.vy	0x87FE17697D0f14A222e8bEf386a0860eCffDD617	
CurveStableSwapNGViews.vy	0x5eeE3091f747E60a045a2E715a4c71e600e31F6E	
CurveStableSwapFactoryNG.vy	0xbC0797015fcFc47d9C1856639CaE50D0e69FbEE8	
CurveStableSwapNG.vy	0xd2002373543Ce3527023C75e7518C274A51ce712	
CurveStableSwapMetaNG.vy	0xd3B17f862956464ae4403cCF829CE69199856e1e	

Polygon:mainnet

File name	Contract deployed on mainnet	Comment
CurveStableSwapNGMath.vy	0xf3A431008396df8A8b2DF492C913706BDB0874ef	evm-version paris
CurveStableSwapNGViews.vy	0x8b3EFBEfa6eD222077455d6f0DCdA3bF4f3F57A6	evm-version paris
CurveStableSwapFactoryNG.vy	0x1764ee18e8B3ccA4787249Ceb249356192594585	evm-version paris
CurveStableSwapNG.vy	0x506F594ceb4E33F5161139bAe3Ee911014df9f7f	evm-version paris
CurveStableSwapMetaNG.vy	0x87FE17697D0f14A222e8bEf386a0860eCffDD617	evm-version paris

Avalanche:mainnet

File name	Contract deployed on mainnet	Comment

File name	Contract deployed on mainnet	Comment
CurveStableSwapNGMath.vy	0xf3A431008396df8A8b2DF492C913706BDB0874ef	evm-version paris
CurveStableSwapNGViews.vy	0x8b3EFBEfa6eD222077455d6f0DCdA3bF4f3F57A6	evm-version paris
CurveStableSwapFactoryNG.vy	0x1764ee18e8B3ccA4787249Ceb249356192594585	evm-version paris
CurveStableSwapNG.vy	0x506F594ceb4E33F5161139bAe3Ee911014df9f7f	evm-version paris
CurveStableSwapMetaNG.vy	0x87FE17697D0f14A222e8bEf386a0860eCffDD617	evm-version paris

Fantom:mainnet

File name	Contract deployed on mainnet	Comment
CurveStableSwapNGMath.vy	0x8b3EFBEfa6eD222077455d6f0DCdA3bF4f3F57A6	evm-version paris
CurveStableSwapNGViews.vy	0x5eeE3091f747E60a045a2E715a4c71e600e31F6E	evm-version paris
CurveStableSwapFactoryNG.vy	0xe61Fb97Ef6eBFBa12B36Ffd7be785c1F5A2DE66b	evm-version paris
CurveStableSwapNG.vy	0xd2002373543Ce3527023C75e7518C274A51ce712	evm-version paris
CurveStableSwapMetaNG.vy	0x686bdb3D24Bc6F3ED89ed3d3B659765c54aC78B4	evm-version paris

BSC:mainnet

File name	Contract deployed on mainnet	Comment
CurveStableSwapNGMath.vy	0x506F594ceb4E33F5161139bAe3Ee911014df9f7f	evm-version paris
CurveStableSwapNGViews.vy	0x1764ee18e8B3ccA4787249Ceb249356192594585	evm-version paris
CurveStableSwapFactoryNG.vy	0xd7E72f3615aa65b92A4DBdC211E296a35512988B	evm-version paris
CurveStableSwapNG.vy	0x604388Bb1159AFd21eB5191cE22b4DeCdEE2Ae22	evm-version paris
CurveStableSwapMetaNG.vy	0x06452f9c013fc37169B57Eab8F50A7A48c9198A3	evm-version paris

Celo:mainnet

File name	Contract deployed on mainnet	Comment
CurveStableSwapNGMath.vy	0xf3A431008396df8A8b2DF492C913706BDB0874ef	evm-version paris
CurveStableSwapNGViews.vy	0x8b3EFBEfa6eD222077455d6f0DCdA3bF4f3F57A6	evm-version paris
CurveStableSwapFactoryNG.vy	0x1764ee18e8B3ccA4787249Ceb249356192594585	evm-version paris
CurveStableSwapNG.vy	0x506F594ceb4E33F5161139bAe3Ee911014df9f7f	evm-version paris
CurveStableSwapMetaNG.vy	0x87FE17697D0f14A222e8bEf386a0860eCffDD617	evm-version paris

1.5 Summary of findings

Severity	# of Findings
Critical	3
High	5
Medium	12
Low	21

ID	Name	Severity	Status
C-1	Rebasing rewards will get stuck on the contract	Critical	Fixed
C-2	`get_virtual_price()` can be manipulated	Critical	Fixed
C-3	Read-only reentrancy in meta_pool with an old base_pool	Critical	Fixed
H-1	Addresses for oracles should be whitelisted	High	Acknowledged
H-2	Incorrect storage update	High	Fixed
H-3	Incorrect work for the base_pool swap	High	Fixed
H-4	An incorrect `rate` update	High	Fixed
H-5	Incorrect rewards distribution	High	Fixed
M-1	Meta_pool doesn't allow base_pools with len(coins) > 3	Medium	Fixed
M-2	Possible DoS of `exchange_received`	Medium	Fixed
M-3	Incorrect parameters passed to fee calculation	Medium	Fixed

M-4	An incorrect oracle update	Medium	Acknowledged
M-5	Dynamic fee not used	Medium	Fixed
M-6	Fee-on-transfer tokens cannot be used as rewards	Medium	Fixed
M-7	LP balance shouldn't be multiplied by virtual_price	Medium	Fixed
M-8	Fees applied twice	Medium	Fixed
M-9	An incorrect sequence of math operations	Medium	Fixed
M-10	Admin balances don't account for potential token rebases	Medium	Fixed
M-11	Rewards `rate` can be set to 0	Medium	Fixed
M-12	`CurveStableSwapNG` and `CurveStableSwapMetaNG` DOS by manual token sent	Medium	Acknowledged
L-1	Unnecessary approve	Low	Fixed
L-2	Unusable constants and parameters	Low	Fixed
L-3	Incorrect comments	Low	Fixed
L-4	Unnecessary checks	Low	Fixed
L-5	Parameters should be restricted	Low	Fixed
L-6	Loops can be simplified	Low	Fixed
L-7	View functions are vulnerable to the read-only reentrancy	Low	Acknowledged
L-8	Input parameters are not validated	Low	Acknowledged
L-9	`_stored_rates` doesn't account for asset types	Low	Acknowledged
L-10	`get_virtual_price` is vulnerable to a donation attack	Low	Fixed

L-11	Incorrect check	Low	Acknowledged
L-12	The killed gauge will return the incorrect rate	Low	Fixed
L-13	Rewards duration should be flexible	Low	Fixed
L-14	An incorrect invariant logged	Low	Fixed
L-15	Weak checks	Low	Acknowledged
L-16	The variable can be used instead of reading from the array	Low	Fixed
L-17	An unnecessary arithmetic operation	Low	Fixed
L-18	It is possible to fire empty events	Low	Fixed
L-19	An incorrect argument name inside the function descriptor	Low	Fixed
L-20	`D_oracle()` One Transaction Manipulation By The Only One Liquidity Provider	Low	Acknowledged
L-21	Pools with low liquidity cannot be used as price oracles	Low	Acknowledged

1.6 Conclusion

During the audit process 3 CRITICAL, 5 HIGH, 12 MEDIUM, and 21 LOW severity findings were spotted. After working on the reported findings, all of them were acknowledged or fixed by the client.
The quality of the code is very high, and there are a lot of comments in the code that simplifies protocol understanding. Test coverage is sufficient, but some edge cases weren't covered before the audit. That is why we recommend keeping in mind that aside from covering project code with general tests and basic user scenarios, it is very important to use a "malicious" mindset and write different attack scenarios in tests (e.g., try to manipulate some of the storage parameters in tests or try to send edge values to some functions).

2.FINDINGS REPORT

2.1 Critical

C-1	Rebasing rewards will get stuck on the contract
Severity	Critical
Status	Fixed in bff1522b

Description

The main problem here is that `stored_balances` does not account for rewards for rebaseable tokens (e.g. stETH):
CurveStableSwapNG.vy#L380
CurveStableSwapMetaNG.vy#L440
This leads to the situation where deposited tokens with accrued rewards cannot be removed from the contract because of the revert on the lines pointed above. The test scenario was sent to the client during the audit. This finding is classified as critical because pools' contracts do not allow upgrades, which means that users' tokens will get stuck on the contract and there will be no possibility to retrieve them.

Recommendation

We recommend updating work with the `stored_balances` so that it will account for possible token balance rebases.

C-2	`get_virtual_price()` can be manipulated
Severity	Critical
Status	Fixed in bff1522b

Description

`get_virtual_price()` can be manipulated by directly transferring tokens to the pools. The thing is that directly transferred tokens can be skimmed via the `exchange_received()` function:
CurveStableSwapNG.vy#L1629
CurveStableSwapMetaNG.vy#L1610
One example of an attack that can make a profit for a hacker is:

1. Directly transfer one of the tokens to a base_pool that was added to a meta_pool.
2. `get_virtual_price()` increases, because D increases and total_supply remains the same.
3. The hacker can call `remove_liquidity_one_coin()` in meta_pool. Due to the increased virtual_price of the base_pool LP token, it will cost a lot less to remove coin[0] from meta_pool.
4. After this, the hacker can call `exchange_received` in the base_pool and return the deposited in (1) funds.
   The test scenario was sent to the client during the audit.
   This finding is classified as critical because many protocols rely on the virtual_price of the pool (even Curve relies on it), and manipulation of the virtual_price is very dangerous.

Recommendation

We recommend updating the design of the exchange_received() and _balances() functions so that they work with donations correctly.

C-3	Read-only reentrancy in meta_pool with an old base_pool
Severity	Critical
Status	Fixed in bff1522b

Description

Old base pools cannot be added to CurveStableSwapFactory. Using an old base pool in meta_pool can lead to read-only reentrancy attacks because of the possible manipulation of the virtual price of a base pool LP token. There is a possible read-only reentrancy attack with a call to `get_virtual_price` in a metapool (At the line CurveStableSwapMetaNG.vy#L457). Virtual price can be incorrectly increased, and that rate can be used during a swap from base pool LP to the second coin in metapool. It will work with old base pools that use ETH (new ones have a reentrancy lock on the `get_virtual_price` function).
This issue has been assigned a CRITICAL severity level because working with old base pools that contain ETH will lead to rate manipulation and funds loss (exchanging tokens using manipulated prices).

Recommendation

We recommend adding checks to the `CurveStableSwapFactoryNG` contract when base pools are added. It shouldn't be possible to add pools paired with ETH.

2.2 High

H-1	Addresses for oracles should be whitelisted
Severity	High
Status	Acknowledged

Description

The current implementation of the Pools Factory allows users to create pools with user-supplied oracles to determine prices of assets:
CurveStableSwapFactoryNG.vy#L531
CurveStableSwapFactoryNG.vy#L657
This allows malicious users to create pools with oracles that can change their returned values. This could lead to imbalanced pools where a malicious user can steal assets via swaps or liquidity removes. However, the pools created by malicious users should not accumulate any liquidity since these pools will not be accepted by the community and LPs in these pools will not be rewarded with CRV tokens.
But there is one more dangerous scenario that can lead to lost value by Curve users. Let's imagine a situation where a new protocol builds an integration with Curve and deploys a stable pool with some custom mechanics, which is allowed because of the user-supplied oracles. But developers didn't pay enough attention to the security of their price oracle, and a hack took place with the manipulation of the price oracle (flashloan manipulation, donation attack, price control in the pool, etc.) set by that team in the pool. In this case, Curve LPs will lose value.
In our opinion, it is impossible to control the quality of price oracles in an automated way (it is impossible to build this type of check inside any function) which is why we recommend adding a whitelist for oracles so the community can assess the quality of new oracles. Moreover, pools with volatile oracles will lead to permanent losses for LPs (if someone decides to create a wETH/wBTC pool with an oracle that sets the price from wETH to wBTC).

Recommendation

We recommend adding a whitelist for oracles' addresses.

Client's commentary

> This is unfortunately a risk that exists for lots of prominent rate-oraclised assets, where a centralised EOA can change the `exchangeRate()` method implementation with no checks. This is a risk that investors into an asset take. The DAO cannot do due diligence for permissionless factory pools, but it can for such pools seeking a gauge (and it does indeed check for obvious risk vectors). So, for now we can do nothing but accommodate arbitrary user-supplied rate oracles.

H-2	Incorrect storage update
Severity	High
Status	Fixed in bff1522b

Description

`stored_balances` is increased by `dx_w_fee` for meta_coin in meta_pool here CurveStableSwapMetaNG.vy#L1065 but actually it should be increased by `dx_w_fee` that returned from the `_meta_add_liquidity` because the actual increase in meta_coin balance will be less than `dx_w_fee` from `_transfer_in` due to possible fees on liquidity addition. This finding is classified as HIGH because the current implementation of the meta_pool will become broken after one call of the exchange_underlying() function (exchange_received will not work after this).

Recommendation

We recommend updating the storage value with the correct value.

H-3	Incorrect work for the base_pool swap
Severity	High
Status	Fixed in bff1522b

Description

Meta_pool incorrectly updates `stored_balances` when a user tries to swap tokens in base_pool from the meta_pool. CurveStableSwapMetaNG.vy#L1087-L1089 In the case of swap base_pool tokens from meta_pool, `stored_balances` shouldn't be updated in `_transfer_in`. Because of this update, the pool will become broken and always revert on the `exchange_received` call.

Recommendation

We recommend removing the ability to swap base_pool tokens from meta_pool or correctly updating meta_pool storage in such cases.

H-4	An incorrect `rate` update
Severity	High
Status	Fixed in bff1522b

Description

`rate` is fetched before epoch update in the `CRV` contract LiquidityGauge.vy#L237-L238 which can lead to an incorrect rate being saved to the storage. This finding is classified as HIGH severity because an incorrect `rate` update will lead to incorrect CRV distribution to users.

Recommendation

We recommend fetching `rate` after the epoch update in the `CRV` contract.

H-5	Incorrect rewards distribution
Severity	High
Status	Fixed in bff1522b

Description

Some of the rewards will be blocked on the contract if they were deposited to the empty gauge (when totalSupply == 0) LiquidityGauge.vy#L318 This happens because `last_update` will be updated nevertheless totalSupply is zero. This finding is classified as HIGH severity since the reward distributor will block some rewards on the contract without a possibility ti retrieve them.

Recommendation

We recommend updating `last_update` only if totalSupply > 0.

2.3 Medium

M-1	Meta_pool doesn't allow base_pools with len(coins) > 3
Severity	Medium
Status	Fixed in bff1522b

Description

The current implementation of the meta_pool doesn't allow base_pools with more than 3 tokens CurveStableSwapMetaNG.vy#L64-L68 but such base_pools can be added in the factory. If meta_pool will be created with base_pool with more than 3 tokens, part of the meta_pool functionality will not work.

Recommendation

We recommend adding a check in the constructor that `BASE_N_COINS < 4`.

M-2	Possible DoS of `exchange_received`
Severity	Medium
Status	Fixed in bff1522b

Description

`exchange_received` can be DoSed if someone sends 1 wei of one of the base_pool tokens to the meta_pool CurveStableSwapMetaNG.vy#L389.

Recommendation

We recommend changing the strict check `dx == _dx` to `dx >= _dx`.

M-3	Incorrect parameters passed to fee calculation
Severity	Medium
Status	Fixed in bff1522b

Description

`xs` is calculated using token balances in the following places:
CurveStableSwapMetaNG.vy#L700
CurveStableSwapMetaNG.vy#L827
CurveStableSwapNG.vy#L589
CurveStableSwapNG.vy#L712.
`ys` is calculated using balances * rates (D / N_COINS ~ balance[i] * rate[i] / PRECISION). As `ys` is calculated with `rates` and `xs` is calculated without them, the fees will be higher than they should be and users will pay more (if rates >> PRECISION e.g. if token decimals < 18).

Recommendation

We recommend multiplying `xs` by `rates`.

M-4	An incorrect oracle update
Severity	Medium
Status	Acknowledged

Description

EMA oracle for `D` is updated with the `D2` value here CurveStableSwapNG.vy#L598 but it should be updated with a slightly different value since `D2` accounts for all fees, but the oracle should be updated with the value that accounts only admin fees.

Recommendation

We recommend calculating the `D3` value that accounts only for admin fees and using this value for the oracle update.

Client's commentary

> The difference is too small to introduce complex gas-consuming computations. For now, it is good enough as it is.

M-5	Dynamic fee not used
Severity	Medium
Status	Fixed in bff1522b

Description

The dynamic fee is calculated here but not accounted for:
CurveStableSwapNG.vy#L714
CurveStableSwapNGViews.vy#L284.

Recommendation

We recommend using a calculated dynamic fee value.

M-6	Fee-on-transfer tokens cannot be used as rewards
Severity	Medium
Status	Fixed in bff1522b

Description

`amount` from transfer is used to update the rate in the liquidity gauge LiquidityGauge.vy#L691. If a fee-on-transfer token is used as a reward token, then some user will not be able to claim rewards until the reward distributor tops up the contract.

Recommendation

We recommend using the exact transferred value instead of the parameter that is passed to the `transfer` call.

M-7	LP balance shouldn't be multiplied by virtual_price
Severity	Medium
Status	Fixed in bff1522b

Description

LP balance from the `_base_calc_token_amounts` shouldn't be multiplied by a virtual_price.
CurveStableSwapNGViews.vy#L116-L118

Recommendation

We recommend removing multiplication by virtual_price of LP balance that returned from the `_base_calc_token_amounts`.

M-8	Fees applied twice
Severity	Medium
Status	Fixed in bff1522b

Description

Fees are already accounted for in the `_base_calc_token_amounts`, so there is no need to account them twice. CurveStableSwapNGViews.vy#L184

Recommendation

We recommend removing fees applying in the `get_dy_underlying` function.

M-9	An incorrect sequence of math operations
Severity	Medium
Status	Fixed in bff1522b

Description

If rates[0] becomes less than 10**18, then `get_dy_underlying` will revert here: CurveStableSwapNGViews.vy#L206.

Recommendation

We recommend updating implementation like this: dy = dy * 10**18 / rates[0].

M-10	Admin balances don't account for potential token rebases
Severity	Medium
Status	Fixed in bff1522b

Description

Admin fees (stored in an array CurveStableSwapMetaNG.vy#L208) don't account for potential slashings. If admin fees are withdrawn first (after the slashing event), then LPs are getting unfairly diluted.
This issue has been assigned a MEDIUM severity level because admin balances don't account for both rebases up and down while slashings are quite rare events (so that rebases down would be outweighed with rebases up).

Recommendation

We recommend adding a comment in the `_balances` function that admin balances don't account for token rebases.

M-11	Rewards `rate` can be set to 0
Severity	Medium
Status	Fixed in bff1522b

Description

There is an issue with a `deposit_reward_token` function defined at the line LiquidityGauge.vy#L680. It is possible to provide quite a big `_epoch` compared to `_amount` being deposited. It can cause the rate to be calculated as 0 here LiquidityGauge.vy#L695 and here LiquidityGauge.vy#L699.
This issue has been assigned a MEDIUM severity level as it will lead to a small amount of reward tokens being stuck on a contract.

Recommendation

We recommend calculating the `rate` value using precision to prevent divisions from leading to zeroes.

M-12	`CurveStableSwapNG` and `CurveStableSwapMetaNG` DOS by manual token sent
Severity	Medium
Status	Acknowledged

Description

In `CurveStableSwapNG` and `CurveStableSwapMetaNG` if 1 wei of any token is sent to an empty pool, `get_D()` fails. Consequently, `add_liquidity()` also fails. This situation can be resolved manually by sending 1 wei of the remaining tokens.
CurveStableSwapNG.vy#L991
CurveStableSwapNG.vy#L1005

Recommendation

We recommend considering the scenario of tokens sent to an empty pool when calculating D.

2.4 Low

L-1	Unnecessary approve
Severity	Low
Status	Fixed in bff1522b

Description

The current implementation of the Factory grants approves tokens to newly created pools, which is unnecessary:
CurveStableSwapFactoryNG.vy#L550-L557
CurveStableSwapFactoryNG.vy#L661.

Recommendation

We recommend removing approves granting from the Factory to newly created pools.

L-2	Unusable constants and parameters
Severity	Low
Status	Fixed in bff1522b

Description

Constants and parameters from the list below are unused and can be removed:
CurveStableSwapFactoryNG.vy#L76
CurveStableSwapMetaNG.vy#L920
CurveStableSwapMetaNG.vy#L1024
CurveStableSwapNG.vy#L315
CurveStableSwapNG.vy#L317
CurveStableSwapNG.vy#L805
CurveStableSwapNGViews.vy#L350
CurveStableSwapNGViews.vy#L664.

Recommendation

We recommend removing unused constants and parameters.

L-3	Incorrect comments
Severity	Low
Status	Fixed in bff1522b

Description

There are several places in the scope where comments are outdated/incorrect:
`Integer array` - CurveStableSwapFactoryNG.vy#L444.
Length of `base_pool_list` instead of `pool_list` here - CurveStableSwapFactoryNG.vy#L89.
duplicated `the` - CurveStableSwapFactoryNG.vy#L477-L478
CurveStableSwapMetaNG.vy#L374
CurveStableSwapMetaNG.vy#L1032
CurveStableSwapMetaNG.vy#L1038
CurveStableSwapNG.vy#L322
CurveStableSwapNG.vy#L367.
Fix spelling - CurveStableSwapNG.vy#L488, CurveStableSwapMetaNG.vy#L547
LiquidityGauge.vy#L88.

Recommendation

We recommend correcting the comments.

L-4	Unnecessary checks
Severity	Low
Status	Fixed in bff1522b

Description

There are several places in the scope with unnecessary checks that can be removed:
CurveStableSwapMetaNG.vy#L456
CurveStableSwapMetaNG.vy#L1012
CurveStableSwapNG.vy#L937
LiquidityGauge.vy#L169.

Recommendation

We recommend removing the unnecessary checks.

L-5	Parameters should be restricted
Severity	Low
Status	Fixed in bff1522b

Description

There are several places in the scope where input parameters should be restricted for correct work:
`len(coins)` should be at least greater or equal to 2:
CurveStableSwapFactoryNG.vy#L490
`_A` and `_ma_exp_time` should be limited:
CurveStableSwapFactoryNG.vy#L519-L533 CurveStableSwapFactoryNG.vy#L642-L659
`_burn_amount` can be 0:
CurveStableSwapMetaNG.vy#L738-L745
`_s` should be restricted:
(https://eips.ethereum.org/EIPS/eip-2) CurveStableSwapMetaNG.vy#L1515 CurveStableSwapNG.vy#L1543
LiquidityGauge.vy#L576
`i` should be restricted within `N_COINS`:
CurveStableSwapNG.vy#L1304 CurveStableSwapNG.vy#L1310 CurveStableSwapNG.vy#L1334
A possible deposit for 0 address:
LiquidityGauge.vy#L418
0-value transfer LiquidityGauge.vy#L497 LiquidityGauge.vy#L511
`_distributor` can be zero address LiquidityGauge.vy#L702

Recommendation

We recommend restricting parameters according to the description.

L-6	Loops can be simplified
Severity	Low
Status	Fixed in bff1522b

Description

Several loops can be simplified:
Th loop can be in range (i, MAX_COINS): CurveStableSwapFactoryNG.vy#L511.
There is no need to get the oracle price for the LP token:
CurveStableSwapMetaNG.vy#L463.

Recommendation

We recommend reducing loop steps to save some gas.

L-7	View functions are vulnerable to the read-only reentrancy
Severity	Low
Status	Acknowledged

Description

Some of the functions are vulnerable to the read-only reentrancy:
CurveStableSwapMetaNG.vy#L1357
CurveStableSwapMetaNG.vy#L1539
CurveStableSwapMetaNG.vy#L1553
CurveStableSwapMetaNG.vy#L1559
CurveStableSwapMetaNG.vy#L1573
CurveStableSwapMetaNG.vy#L1579
CurveStableSwapMetaNG.vy#L1618
CurveStableSwapNG.vy#L1315
CurveStableSwapNG.vy#L1567
CurveStableSwapNG.vy#L1581
CurveStableSwapNG.vy#L1595
CurveStableSwapNG.vy#L1637.

If another protocol decides to use these functions as price sources, then it can be attacked via the read-only reentrancy if the pool contains hookable tokens.

Recommendation

We recommend adding a `nonreentrant` lock for these functions.

L-8	Input parameters are not validated
Severity	Low
Status	Acknowledged

Description

The current implementation of the Factory contract doesn't properly validate input parameters for the `add_pool` function CurveStableSwapFactoryNG.vy#L719-L725 which allows admins to add meta_pool as base_pool by mistake.

Recommendation

We recommend reading parameters from the pool instead of passing them to the function.

L-9	`_stored_rates` doesn't account for asset types
Severity	Low
Status	Acknowledged

Description

`_stored_rates` doesn't account for asset types, so an asset with type 0 and set by a mistake oracle will work as type 1:
CurveStableSwapNG.vy#L415
CurveStableSwapMetaNG.vy#L479.

Recommendation

We recommend accounting for asset type or removing type 1.

L-10	`get_virtual_price` is vulnerable to a donation attack
Severity	Low
Status	Fixed in bff1522b

Description

The current implementation of `get_virtual_price` allows a malicious user to directly transfer some funds to the contract and increase `get_virtual_price`. This behavior can be used in a complex hack if it makes economic sense to lose some value by a direct transfer of funds to the contract and use the manipulated value of `get_virtual_price` in another protocol to steal more assets that were transferred to the protocol.

Recommendation

We recommend adding this information to the documentation so protocols that will decide to integrate with stable pools will be aware of this risk.

L-11	Incorrect check
Severity	Low
Status	Acknowledged

Description

The check used here LiquidityGauge.vy#L236 leads to the update of `inflation_params` on every call of `_checkpoint`.

Recommendation

We recommend changing this check to:
`if prev_future_epoch <= block.timestamp:`.

L-12	The killed gauge will return the incorrect rate
Severity	Low
Status	Fixed in bff1522b

Description

`rate` is not updated in the killed gauge LiquidityGauge.vy#L242-L243 which will lead to a situation where the killed gauge returns a non-zero rate.

Recommendation

We recommend updating `self.inflation_params` for the killed gauge.

L-13	Rewards duration should be flexible
Severity	Low
Status	Fixed in bff1522b

Description

In the current implementation of the Liquidity Gauge, rewards can be set for one week only LiquidityGauge.vy#L691. Some of the financial teams of the protocols plan budget for a month, so it will be more convenient for them to set reward duration as one month.

Recommendation

We recommend giving more flexibility to reward duration.

L-14	An incorrect invariant logged
Severity	Low
Status	Fixed in bff1522b

Description

There is an issue at the lines:
CurveStableSwapNG.vy#L615, CurveStableSwapMetaNG.vy#L733, CurveStableSwapNG.vy#L730, and CurveStableSwapMetaNG.vy#L847. `D1` is logged as an invariant even if it were recalculated to `D2` (accounting for applied fees).

Recommendation

We recommend logging `D1` or `D2` depending on whether the fees were applied or not.

L-15	Weak checks
Severity	Low
Status	Acknowledged

Description

There are weak checks at line CurveStableSwapNG.vy#L414 and CurveStableSwapMetaNG.vy#L476. There is no need in these checks because if there weren't a revert on a call to the oracle, the response length would always be equal to 32.

Recommendation

We recommend introducing more secure checks that can check for the returned value from the oracle.

L-16	The variable can be used instead of reading from the array
Severity	Low
Status	Fixed in bff1522b

Description

There is an `xp_j` variable defined here - CurveStableSwapNG.vy#L1177. This variable can be used in the following calculations - CurveStableSwapNG.vy#L1181 and CurveStableSwapNG.vy#L1184 instead of reading array member `xp[j]`. The same issue can be found at line CurveStableSwapMetaNG.vy#L1233 and CurveStableSwapMetaNG.vy#L1236.

Recommendation

We recommend using the `xp_j` value on the mentioned lines instead of accessing the `xp[j]` array member.

L-17	An unnecessary arithmetic operation
Severity	Low
Status	Fixed in bff1522b

Description

There is an issue at line LiquidityGauge.vy#L254. There is no need to divide and then multiply the `prev_week_time` value by `WEEK` as it is done inside the GaugeController contract.

Recommendation

We recommend removing unnecessary arithmetic operations.

L-18	It is possible to fire empty events
Severity	Low
Status	Fixed in bff1522b

Description

There is an issue at line LiquidityGauge.vy#L435 and LiquidityGauge.vy#L436. It is possible to call the `deposit` function with `_value` equal to 0 which will lead to the `Deposit` and `Transfer` events being emitted.

Recommendation

We recommend emitting the mentioned events only if `_value` isn't equal to 0.

L-19	An incorrect argument name inside the function descriptor
Severity	Low
Status	Fixed in bff1522b

Description

There is an incorrect argument name used at the lines: CurveStableSwapNG.vy#L53, CurveStableSwapMetaNG.vy#L54 and CurveStableSwapMetaNG.vy#L56. `dx` should be used instead of `dy`.

Recommendation

We recommend changing the argument name inside the mentioned declarations.

L-20	`D_oracle()` One Transaction Manipulation By The Only One Liquidity Provider
Severity	Low
Status	Acknowledged

Description

If there is only one liquidity provider in `CurveStableSwapNG` or `CurveStableSwapMetaNG` at the moment, it's possible to get two different values of `D_oracle()` in one transaction by the following algorithm:

1. Use old values based on previous block info.
2. Remove all liquidity. `total_supply` becomes equal to 0.
3. Add new liquidity to an empty pool; `last_D_packed` is overwritten and can be used immediately by `D_oracle()`.
   CurveStableSwapNG.vy#L1340
   CurveStableSwapNG.vy#L605

Recommendation

We recommend not overwriting `last_D_packed` when adding liquidity to an empty pool, but using a general mechanism for the moving average.

L-21	Pools with low liquidity cannot be used as price oracles
Severity	Low
Status	Acknowledged

Description

There is a risk of usage pools with low liquidity or with low trading volume as price oracles. Protocols that will be integrated with Curve StablePools and use prices from them as price oracles should be aware of these risks and check pool parameters (liquidity, trading volume) during their usage.

Recommendation

We recommend adding a warning about the usage of stale pools or pools with low liquidity as price oracles to the documentation.

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build opensource solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

Contacts

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://twitter.com/mixbytes