EYWA CLP SECURITY AUDIT REPORT

Oct 10, 2024

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. 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

Eywa CLP is a cross-chain trading and yield protocol that addresses the issue of fragmented liquidity through the use of Curve Finance's deep liquidity pools. The protocol connects all Curve liquidity from different blockchains into a single entity, creating a global market, and enabling low-slippage cross-chain swaps.

1.4 Project Dashboard

Project Summary

Title	Description
Client	Eywa
Project name	CLP
Timeline	14.06.2024 - 13.08.2024
Number of Auditors	4

Project Log

Date	Commit Hash	Note
14.06.2024	d68ba027ff19e927d64de123b2b02f15a43f8214	Commit for the audit
09.08.2024	67f35d703c089304cfdbf161ab6cfa730a020d86	Commit for the re-audit

Project Scope

The audit covered the following files:

File name	Link
contracts/adapters/stable3/PoolAdapter.sol	PoolAdapter.sol
contracts/adapters/crypto1/PoolAdapter.sol	PoolAdapter.sol
contracts/adapters/crypto3/PoolAdapter.sol	PoolAdapter.sol
contracts/adapters/stable4/PoolAdapter.sol	PoolAdapter.sol

File name	Link
contracts/adapters/stable1/PoolAdapter.sol	PoolAdapter.sol
contracts/adapters/crypto2/PoolAdapter.sol	PoolAdapter.sol
contracts/adapters/meta1/PoolAdapter.sol	PoolAdapter.sol
contracts/adapters/stable2/PoolAdapter.sol	PoolAdapter.sol
contracts/utils/Utils.sol	Utils.sol
contracts/utils/BytesLib.sol	BytesLib.sol
contracts/utils/Typecast.sol	Typecast.sol
contracts/utils/RequestIdLib.sol	RequestIdLib.sol
contracts/utils/FrontHelper.sol	FrontHelper.sol
contracts/RouterV2.sol	RouterV2.sol
contracts/SynthERC20.sol	SynthERC20.sol
contracts/Treasury.sol	Treasury.sol
contracts/EndPoint.sol	EndPoint.sol
contracts/BaseRouter.sol	BaseRouter.sol
contracts/BalancerTreasury.sol	BalancerTreasury.sol
contracts/AddressBook.sol	AddressBook.sol
contracts/SynthFactory.sol	SynthFactory.sol
contracts/Whitelist.sol	Whitelist.sol
contracts/FeesTreasury.sol	FeesTreasury.sol
contracts/SynthesisV2.sol	SynthesisV2.sol

File name	Link
contracts/UnifiedRouterV2.sol	UnifiedRouterV2.sol
contracts/vp/VirtualPriceSender.sol	VirtualPriceSender.sol
contracts/vp/VirtualPriceReceiver.sol	VirtualPriceReceiver.sol
contracts/vp/VirtualPriceReceiverV2.sol	VirtualPriceReceiverV2.sol
contracts/PortalV2.sol	PortalV2.sol
contracts/ThirdPartySynthAdapter.sol	ThirdPartySynthAdapter.sol
contracts/OpsRegistrar.sol	OpsRegistrar.sol

Deployments

Ethereum:mainnet

File name	Contract address	Comment
AddressBook.sol	0x78C973...7e461005	
PortalV2.sol	0xac8f44...0a0ffcbe	
SynthFactory.sol	0x1B8607...7af69178	
UnifiedRouterV2.sol	0xA2A786...94585c74	
OpsRegistrar.sol	0x5571E3...4A8fDbB9	
Whitelist.sol	0x5cF21A...fAd6bB8F	
FeesTreasury.sol	0x440067...cd292845	
VirtualPriceSender.sol	0xfA843a...67142541	
PoolAdapter.sol	0x4a7942...0d8a3BB0	lp: 0x6c3f90...bde6e490, n: 3

File name	Contract address	Comment
PoolAdapter.sol	0x2B41Ff...63124227	
PoolAdapter.sol	0x170f58...50701450	PoolAdapterCrypto

BSC:mainnet

File name	Contract address	Comment
AddressBook.sol	0x78C973...7e461005	
PortalV2.sol	0xac8f44...0a0ffcbe	
SynthFactory.sol	0x1B8607...7af69178	
UnifiedRouterV2.sol	0xA2A786...94585c74	
OpsRegistrar.sol	0x5571E3...4A8fDbB9	
Whitelist.sol	0x5cF21A...fAd6bB8F	
FeesTreasury.sol	0x440067...cd292845	
VirtualPriceSender.sol	0xfA843a...67142541	
PoolAdapter.sol	0x808619...8bAb8Bea	PoolAdapterStableNg

Polygon:mainnet

File name	Contract address	Comment
AddressBook.sol	0x78C973...7e461005	
PortalV2.sol	0xac8f44...0a0ffcbe	
SynthFactory.sol	0x1B8607...7af69178	
UnifiedRouterV2.sol	0xA2A786...94585c74	

File name	Contract address	Comment
OpsRegistrar.sol	0x5571E3...4A8fDbB9	
Whitelist.sol	0x5cF21A...fAd6bB8F	
FeesTreasury.sol	0x440067...cd292845	
VirtualPriceSender.sol	0xfA843a...67142541	
PoolAdapter.sol	0xb6dB0c...5fcB99C2	Aave, n: 3
PoolAdapter.sol	0x8933f2...dCC43792	stableNG
PoolAdapter.sol	0x0D1CBc...b2A250F5	
PoolAdapter.sol	0x7ca38D...8cDCda62	zap

Avalanche:mainnet

File name	Contract address	Comment
AddressBook.sol	0x78C973...7e461005	
PortalV2.sol	0xac8f44...0a0ffcbe	
SynthFactory.sol	0x1B8607...7af69178	
UnifiedRouterV2.sol	0xA2A786...94585c74	
OpsRegistrar.sol	0x5571E3...4A8fDbB9	
Whitelist.sol	0x5cF21A...fAd6bB8F	
FeesTreasury.sol	0x440067...cd292845	
VirtualPriceSender.sol	0xfA843a...67142541	
PoolAdapter.sol	0x20a833...a79515b9	Aave, n: 3

File name	Contract address	Comment
PoolAdapter.sol	0x7388E9...12d55826	
PoolAdapter.sol	0x31F88E...d00160c9	ZAP

Avalanche:mainnet

File name	Contract address	Comment
AddressBook.sol	0x78C973...7e461005	
PortalV2.sol	0xac8f44...0a0ffcbe	
SynthFactory.sol	0x1B8607...7af69178	
UnifiedRouterV2.sol	0xA2A786...94585c74	
OpsRegistrar.sol	0x5571E3...4A8fDbB9	
Whitelist.sol	0x5cF21A...fAd6bB8F	
FeesTreasury.sol	0x440067...cd292845	
VirtualPriceSender.sol	0xfA843a...67142541	
PoolAdapter.sol	0x3ed9eB...5178a927	n: 3
PoolAdapter.sol	0x66A2de...68188610	stableNG

Arbitrum:mainnet

File name	Contract address	Comment
AddressBook.sol	0x78C973...7e461005	
PortalV2.sol	0xac8f44...0a0ffcbe	
SynthFactory.sol	0x1B8607...7af69178	

File name	Contract address	Comment
UnifiedRouterV2.sol	0xA2A786...94585c74	
OpsRegistrar.sol	0x5571E3...4A8fDbB9	
Whitelist.sol	0x5cF21A...fAd6bB8F	
FeesTreasury.sol	0x440067...cd292845	
VirtualPriceSender.sol	0xfA843a...67142541	**Deployment differs from the last commit**
PoolAdapter.sol	0x59dD71...a5A2f502	n: 2
PoolAdapter.sol	0xaB54E3...B22F5133	stableNG
PoolAdapter.sol	0x1D7294...d55c1f62	

Base:mainnet

File name	Contract address	Comment
AddressBook.sol	0x78C973...7e461005	
PortalV2.sol	0xac8f44...0a0ffcbe	
SynthFactory.sol	0x1B8607...7af69178	
UnifiedRouterV2.sol	0xA2A786...94585c74	
OpsRegistrar.sol	0x5571E3...4A8fDbB9	
Whitelist.sol	0x5cF21A...fAd6bB8F	
FeesTreasury.sol	0x440067...cd292845	
VirtualPriceSender.sol	0xfA843a...67142541	
PoolAdapter.sol	0x003FBE...3b5d3573	n: 4

Gnosis:mainnet

File name	Contract address	Comment
AddressBook.sol	0x78C973...7e461005	
PortalV2.sol	0xac8f44...0a0ffcbe	
SynthFactory.sol	0x1B8607...7af69178	
UnifiedRouterV2.sol	0xA2A786...94585c74	
OpsRegistrar.sol	0x5571E3...4A8fDbB9	
Whitelist.sol	0x5cF21A...fAd6bB8F	
FeesTreasury.sol	0x440067...cd292845	
VirtualPriceSender.sol	0xfA843a...67142541	
PoolAdapter.sol	0x5a072a...E6C41d85	lp: 0x1337be...a27963ec, n: 3

Mantle:mainnet

File name	Contract address	Comment
AddressBook.sol	0x78C973...7e461005	
PortalV2.sol	0xac8f44...0a0ffcbe	
SynthFactory.sol	0x1B8607...7af69178	
UnifiedRouterV2.sol	0xA2A786...94585c74	
OpsRegistrar.sol	0x5571E3...4A8fDbB9	
Whitelist.sol	0x5cF21A...fAd6bB8F	
FeesTreasury.sol	0x440067...cd292845	

File name	Contract address	Comment
VirtualPriceSender.sol	0xfA843a...67142541	

Blast:mainnet

File name	Contract address	Comment
AddressBook.sol	0x78C973...7e461005	
PortalV2.sol	0xac8f44...0a0ffcbe	
SynthFactory.sol	0x1B8607...7af69178	
UnifiedRouterV2.sol	0xA2A786...94585c74	
OpsRegistrar.sol	0x5571E3...4A8fDbB9	
Whitelist.sol	0x5cF21A...fAd6bB8F	
FeesTreasury.sol	0x440067...cd292845	
VirtualPriceSender.sol	0xfA843a...67142541	

Linea:mainnet

File name	Contract address	Comment
AddressBook.sol	0x78C973...7e461005	
PortalV2.sol	0xac8f44...0a0ffcbe	
SynthFactory.sol	0x1B8607...7af69178	
UnifiedRouterV2.sol	0xA2A786...94585c74	
OpsRegistrar.sol	0x5571E3...4A8fDbB9	
Whitelist.sol	0x5cF21A...fAd6bB8F	

File name	Contract address	Comment
FeesTreasury.sol	0x440067...cd292845	
VirtualPriceSender.sol	0xfA843a...67142541	

Taiko:mainnet

File name	Contract address	Comment
AddressBook.sol	0x78C973...7e461005	
PortalV2.sol	0xac8f44...0a0ffcbe	
SynthFactory.sol	0x1B8607...7af69178	
UnifiedRouterV2.sol	0xA2A786...94585c74	
OpsRegistrar.sol	0x5571E3...4A8fDbB9	
Whitelist.sol	0x5cF21A...fAd6bB8F	
FeesTreasury.sol	0x440067...cd292845	
VirtualPriceSender.sol	0xfA843a...67142541	

Fantom:mainnet

File name	Contract address	Comment
AddressBook.sol	0x78C973...7e461005	
PortalV2.sol	0xac8f44...0a0ffcbe	
SynthFactory.sol	0x1B8607...7af69178	
UnifiedRouterV2.sol	0xA2A786...94585c74	
OpsRegistrar.sol	0x5571E3...4A8fDbB9	

File name	Contract address	Comment
Whitelist.sol	0x5cF21A...fAd6bB8F	
FeesTreasury.sol	0x440067...cd292845	
VirtualPriceSender.sol	0xfA843a...67142541	
VirtualPriceReceiver.sol	0x191a24...27B77b97	xCRVUSDC
VirtualPriceReceiver.sol	0xb756a5...12d47578	x3CRYPTO
VirtualPriceReceiver.sol	0x924417...45F36356	xCRVUSDT
VirtualPriceReceiver.sol	0xd562AC...99B74793	xSTABLE
Locker.sol	0x7e0746...eefd4f69	
PoolAdapter.sol	0xf6f3e6...8DDB2217	stableNG

1.5 Summary of findings

Severity	# of Findings
Critical	0
High	3
Medium	6
Low	14

ID	Name	Severity	Status
H-1	DOS of pool adapters via positive allowance	High	Fixed
H-2	Signature verification bypass	High	Fixed
H-3	`BURN_UNLOCK_CODE` does not validate `p.tokenIn`	High	Fixed
M-1	User incurs fee loss when using `BURN_UNLOCK_CODE` with a `NotSet` token status	Medium	Acknowledged
M-2	Risk of fee loss due to transactions through a paused router	Medium	Acknowledged
M-3	Potential loss of funds if `msg.value` exceeds `amountIn + executionPrice`	Medium	Fixed
M-4	Lack of user control over maximum bridge fee	Medium	Acknowledged
M-5	Transaction DOS via `permit()` front-running	Medium	Fixed
M-6	Unexpected `PERMIT_CODE` in `RouterV2`	Medium	Fixed
L-1	Redundant call with zero`executionPrice`	Low	Fixed

L-2	`RouterV2.receive()` function access to be restricted	Low	Fixed
L-3	Inability of Synthesis to call `burnWithAllowanceDecrease()` or `mintWithAllowanceIncrease()` functions	Low	Fixed
L-4	Unused events	Low	Fixed
L-5	Unused code	Low	Fixed
L-6	Using `Owanble` instead of `Ownable2Step`	Low	Fixed
L-7	Typos	Low	Fixed
L-8	Redundant `DifferentDecimalsAdapter.withdraw()` function to be removed	Low	Fixed
L-9	The `nonReentrant` modifier should occur `before` all other modifiers	Low	Fixed
L-10	Slippage value doesn't account for `bridgeFee`	Low	Acknowledged
L-11	`@TODO` flag in Utils library	Low	Acknowledged
L-12	Incompatibility with fee-on-transfer or rebasing tokens	Low	Acknowledged
L-13	Vulnerability to withdrawal of stuck funds in certain contracts	Low	Acknowledged
L-14	Emergency Handling for Cross-Chain Operations	Low	Acknowledged

1.6 Conclusion

The audit considered the following key aspects of the system, potential attack vectors, and failure points:

1. **Synth tokens**: Synth tokens (both `DefaultSynth` and `CustomSynth`) are created through the `SynthFactory` contract via `create2`. During the audit process, we confirmed that malicious actors cannot disrupt the token creation by sending funds to a synth token's precomputed address. Additionally, when tokens are locked, users are guaranteed to receive the synth token set by the admin in the `Synthesis` contract.

2. **Locked Assets**: The audit established that funds are secure within the `PortalV2` contract due to its robust implementation, which safeguards against DoS attacks. Users are allowed to lock tokens that were previously whitelisted by the admin. It is worth noting that admins retain the authority to remove tokens from the whitelist by setting the token status to `NotSet`, thereby disallowing the unlocking of associated funds.

3. **Cross-chain message handling**: Cross-chain data can only be sent by the `router` contract. All messages from `chain A` undergo an authorization check in `chain B`, allowing messages only from authorized contracts. The unique `requestId` cannot be tampered with, as it is created using the router's `nonce` and `block.chainId`. A message with the same `requestId` can only be used once. If the message fails in `chain B`, admins have the right to call emergency operations to return the tokens locked/burned in `chain A` back to the user. It is also worth noting that a user can potentially DoS cross-chain emergency operations if they manage to bypass signature verification.

4. **User flow**: The user is permitted call the `start()` function only with a predefined list of operations set by the admin. Moreover, the user must obtain a signature from the `accountant` to make a `start()` call. This procedure narrows the scope of potential user-initiated attacks. Additionally, the admin possesses the authority to set the fee value for minting synth tokens and unlocking tokens, with the upper limit capped at `100%`.

5. **Access restriction**: All functions not intended to be called by the user are properly restricted. However, the current backend configuration implies a high trust assumption. For example, the backend can sign a separate transaction with `EMERGENCY_UNLOCK_CODE` or `EMERGENCY_MINT_CODE` to cancel any pending operation awaiting the second step on the second chain.

6. **Code style and architecture**: Overall, the codebase is of high quality. There are a few places with unused code that are not relevant to the current scope. Most of the code is commented. The protocol has well-described documentation that explains most user actions.

2.FINDINGS REPORT

2.1 Critical

Not Found

2.2 High

H-1	DOS of pool adapters via positive allowance
Severity	High
Status	Fixed in 67f35d70

Description

All pool adapters have public `swap()` methods. A hacker can directly call `swap()` to invoke `safeIncreaseAllowance()` and then leave this allowance unspent due to `minAmountOut > minDy`:

```
SafeERC20.safeIncreaseAllowance(erc20Impl, pool, amountIn);

uint256 minDy = poolImpl.get_dy(uint256(i), uint256(j), amountIn);
if (minAmountOut > minDy) {
  SafeERC20.safeTransfer(erc20Impl, emergencyTo, amountIn);
  return 0;
}
```

PoolAdapter.sol#L119

Some tokens (such as USDT, for example) revert a positive `approve(amount)` if there is already a non-zero `allowance`. This scenario was not accounted for in older versions of OpenZeppelin's `safeIncreaseAllowance()`.

The project uses outdated version containing the bug, making it possible for a hacker to disrupt pool adapters for certain tokens like USDT.

In later versions of OpenZeppelin, the bug was fixed with `safeIncreaseAllowance()` setting the initial allowance to zero before setting it to a new positive value.

Recommendation

We recommend updating OpenZeppelin dependencies to the latest versions.

Client's commentary

> fixed
> e5f4de9f2d2bf756bfad7a33025dceb68aa0eaba, 438bebd9e61f3d9e6e99b6872eac444b54dd5e22

H-2	Signature verification bypass
Severity	High
Status	Fixed in 67f35d70

Description

- BaseRouter.sol#L271-L272

The `BaseRouter._getRawData()` function concatenates operations and parameters without separators. This allows a hacker to feed operations to the `RouterV2.start()` with different parameters while maintaining the same signature.

For example, given the operations `[op1, op2]` and parameters `[[a, b], [c, d]]`, the `BaseRouter._checkSignature()` method cannot distinguish between `[[a, b], [c, d]]` and `[[a], [b, c, d]]` or `[[a, b, c], [d]]`, because `_getRawData()` will always concatenate parameters into a single stream of `[a, b, c, d]`.

Moreover, there will also be no errors with unpacking the parameters, because `abi.decode()` does not revert when provided with excess data; it discards extra bytes.

Consequently, while the backend may sign legitimate data, an attacker could reorganize this data in such a way that the operations will execute with entirely different arguments. Although this manipulation will eventually cause a revert due to missing arguments in the final operations, an attacker could potentially extract funds prior to the revert by initiating a swap with slippage, directing funds to an emergency address controlled by the attacker.

Recommendation

We recommend introducing separators between concatenated operations and parameters to ensure they are distinctly identifiable.

Client's commentary

> fixed
> a36aee7d38caa0cf44e1ea91d10061d183e85263

H-3	`BURN_UNLOCK_CODE` does not validate `p.tokenIn`
Severity	High
Status	Fixed in 67f35d70

Description

- RouterV2.sol#L120

The `BURN_UNLOCK_CODE` function does not validate `p.tokenIn`, which can lead to the theft of funds. If the backend is compromised, an attacker could insert a malicious contract as the `p.tokenIn` parameter into the `BURN_UNLOCK_CODE` operation. This would result in the `possibleAdapter` being set to the fake `p.tokenIn`, enabling all subsequent calls to it to proceed without any monetary cost. This vulnerability could allow an attacker to unlock any token on subsequent chains, provided it is returned from the `possibleAdapter.originalToken()` function.

Recommendation

We recommend implementing strict validation of the `p.tokenIn` parameter to confirm its authenticity as a legitimate and trusted contract. Additionally, we recommend implementing security mechanisms such as whitelisting or signature verification to authenticate the origin and integrity of the `p.tokenIn` parameter. This will help prevent unauthorized contracts from being exploited through `BURN_UNLOCK_CODE` and other sensitive operations.

Client's commentary

> fixed
> 119d87a631fe7d3422d49be76fa2b7b974669dbe

2.3 Medium

M-1	User incurs fee loss when using `BURN_UNLOCK_CODE` with a `NotSet` token status
Severity	Medium
Status	Acknowledged

Description

A user's funds may get stuck when using `BURN_UNLOCK_CODE` for a `NotSet` token.

For example, consider the following scenario:

1. A user locks `1000e18` LP tokens on chain A and mints corresponding synthLp tokens on chain B.
2. As time progresses, the `WhitelistV2` contract's owner sets the LP token's status on chain A to `NotSet`.
3. The user decides to unlock their LP tokens and `BURN_UNLOCK_CODE` from chain B to A.
4. While the "burn" operation is successful on chain B, it fails on chain A due to the requirement that the output token must be set as `InOut`:

```
require(
    IWhitelist(whitelist).tokenState(otoken)
        == uint8(IWhitelist.TokenState.InOut),
    "Portal: token must be whitelisted");
```

PortalV2.sol#L87

Therefore, the user loses the commission for the cross-chain transaction.

Recommendation

We recommend adding a check (similar to the one in the Portal contract) during operations with synthetic tokens in the Synthesis contract to ensure that the token has been added to the whitelist in the 'Whitelist' contract.

Client's commentary

> not an issue
> Whitelist contract will be controlled by DAO once it will be released. For now it's controlled by operator mulisig. There are many settings that need to be in a consistent state.

M-2	Risk of fee loss due to transactions through a paused router
Severity	Medium
Status	Acknowledged

Description

The `UnifiedRouterV2` contract can be paused. If the user's path includes a paused router, their transaction can fail, and they might simply lose money on fees.

Recommendation

We recommend notifying users on the front-end when contracts are paused.

Client's commentary

> not an issue
> For emergency cases router can be paused by operator. Front-end will notify user that's some cross-chain swaps are impossible at the moment.

M-3	Potential loss of funds if `msg.value` exceeds `amountIn + executionPrice`
Severity	Medium
Status	Fixed in 67f35d70

Description

If a user conducts a cross-chain operation with `WRAP_CODE`, they transfer native currency to the router and, in return, receive WETH. At the same time, the user pays an `executionPrice` for the cross-chain transaction.

However, there's a risk that users might input a `msg.value` significantly higher than needed, leading to the excess `native currency` being stuck on the router.

RouterV2.sol#L213

Recommendation

We recommend adding a `require` statement to check that `msg.value` is not more than `executionPrice + amountIn` when using the `WRAP_CODE` operation.

Client's commentary

> fixed
> 00d2e99f7d15ba42eb963ea75f2136d30ce538a9

M-4	Lack of user control over maximum bridge fee
Severity	Medium
Status	Acknowledged

Description

- PortalV2.sol#L89
- Whitelist.sol#L90

The `PortalV2.unlock()` function charges the user a fee specified in the Whitelist contract, which can be increased by the admin up to 100%, affecting transactions already in progress based on lower fees. Currently, the user cannot set a threshold for the maximum fee, which can lead to situations where a user initiates a bridge transaction expecting a nominal fee, but the admin - without ill intent - increases the fee, resulting in the user receiving less money than anticipated.

Recommendation

There are various ways to address this issue. One option is to empower users to set a maximum fee before sending a transaction. Another option is to implement a mechanism to update the bridge fee with a notice period, such as one day, to alert users of the change.

Client's commentary

> Front-end notifies user about comission.

M-5	Transaction DOS via `permit()` front-running
Severity	Medium
Status	Fixed in 67f35d70

Description

The `permit()` data, once submitted, is publicly accessible in the mempool, allowing anyone to execute the permit by replicating the transaction arguments. Once `permit()` has been called, the second call with identical parameters will revert.

In a scenario where a signed transaction includes `PERMIT_CODE`, a malicious actor could frontrun and "activate" this permit, bypassing the router's `start()` function. As a result, the legitimate user's `start()` transaction would fail:

- RouterV2.sol#L99-L109

Reference:

- https://www.trust-security.xyz/post/permission-denied

Recommendation

We recommended using the `try/catch` pattern for permit operations to prevent reverts.

Client's commentary

> fixed
> ba4ea5fa8174ef0a8e3818ef142e8639e3de0e71

M-6	Unexpected `PERMIT_CODE` in `RouterV2`
Severity	Medium
Status	Fixed in 67f35d70

Description

The `PERMIT_CODE` is assumed to always be the initial operation in the list. As a result, the `_checkTo()` method does not anticipate this code to follow and consequently returns zero.

- RouterV2.sol#L274

Recommendation

We recommend ensuring that the `PERMIT` operation is either the first in the sequence or not included at all.

Client's commentary

> fixed
> 8fb564c780d619a30c64524725fd7e898d67253e

2.4 Low

L-1	Redundant call with zero`executionPrice`
Severity	Low
Status	Fixed in 67f35d70

Description

The `executionPrice` value may be set to zero during non cross-chain calls. However, it still transfers a 0 value to the accountant when `executionPrice` is zero.

RouterV2.sol#L229-L233

Recommendation

We don't recommend making a zero value call when `executionPrice` is zero.

```
    function _proceedFees(uint256 executionPrice, address accountant)
    internal
    virtual
    override {
++      if (executionPrice != 0) {
            require(msg.value >= executionPrice,
                "Router: invalid amount");
            (bool sent, ) = accountant.call{ value: executionPrice }("");
            require(sent, "Router: failed to send Ether");
        }
        emit FeePaid(msg.sender, accountant, executionPrice);
    }
```

Client's commentary

> fixed
> 5587f5c7c8c873f1f85167290b0f2ab9fb7d9fc5

L-2	`RouterV2.receive()` function access to be restricted
Severity	Low
Status	Fixed in 67f35d70

Description

The `RouterV2` contract is capable of receiving `Ether` from any user due a payable `receive()` function. This could lead to unintended fund losses for users. The only scenario where transferring `Ether` to the router is intended is when a user employs the `WRAP_CODE`.

Recommendation

We recommend introducing a require statement ensuring that only the `WETH` contract can transfers funds to the router.

```
    receive() external payable {
++      require(msg.sender
            == address(WETH), "Invalid sender");
    }
```

RouterV2.sol#L48

Client's commentary

> fixed
> 4185277891708e5cf4c8c4fb880373b6084e6a5f, 67f35d703c089304cfdbf161ab6cfa730a020d86

L-3	Inability of Synthesis to call `burnWithAllowanceDecrease()` or `mintWithAllowanceIncrease()` functions
Severity	Low
Status	Fixed in 67f35d70

Description

While the `SynthesisV2` contract maintains the ownership of the `SynthERC20` token, it lacks the functionality to call `burnWithAllowanceDecrease()` and `mintWithAllowanceIncrease()`.

SynthERC20.sol#L61-L68

SynthERC20.sol#L74-L84

Recommendation

We recommend either integrating these functions into the `SynthesisV2` contract's functionality or removing them from `SynthERC20`.

Client's commentary

> fixed
> 5bea6d7b3aee8e34019f337303bcf3b66f76d4ef

L-4	Unused events
Severity	Low
Status	Fixed in 67f35d70

Description

The `UnifiedRouterV2` contract includes an unused event, `PoolAdapterSet`, which is not emitted when the operator sets the pool adapter.

UnifiedRouterV2.sol#L12

In the `DifferentDecimalsAdapter` contract, the `CapSet` event is not used.

DifferentDecimalsAdapter.sol#L40-L42

Recommendation

We recommend emitting these events.

Client's commentary

> fixed
> 2672f0606e7d740aa9831f093d49f3a69ee34187

L-5	Unused code
Severity	Low
Status	Fixed in 67f35d70

Description

The `Whitelist` contract contains numerous instances of unused code related to pools (this code is never used in the current scope).

Whitelist.sol#L20-L22

Whitelist.sol#L25

Whitelist.sol#L52-L62

Whitelist.sol#L74-L82

Whitelist.sol#L103-L119

Whitelist.sol#L128-L133

Additionally, in `PoolAdapterAave`, the `isUnderlying()` function is present but commented out, indicating it is not in use.

PoolAdapter.sol#L180-L189

Recommendation

We recommend removing the unused code.

Client's commentary

> fixed
> b22c8faeb1389ff795d7e047e931d5eb813cdd5e, e4a57196672a96233a54531491e246b9a28b36c0, 9a71e9ee3e11ef21f349fedc706ff73f1ef222b2

L-6	Using `Owanble` instead of `Ownable2Step`
Severity	Low
Status	Fixed in 67f35d70

Description

Some contracts like `AddressBook`, `DifferentDecimalsAdapter`, `FeesTreasury`, `PortalV2`, `SynthERC20`, `ThirdPartySynthAdapter`, `Whitelist` currently inherit from `Ownable` by `OpenZeppelin`.

It's safer to inherit from `Ownable2Step` to avoid a situation when the contract ownership could be inadvertently transferred to an invalid address.

Recommendation

We recommend inheriting from `Ownable2Step` by `OpenZeppelin` or using a multisig or DAO as the contract owner.

Client's commentary

> fixed for FeesTreasury
> 2528b9df98d9ea39de12faa9cd3bb2fa1b0ef048

L-7	Typos
Severity	Low
Status	Fixed in 67f35d70

Description

In the `Utils` library, there is a typo on L262.

```
    require(key.length
            >= 67, "key lenggh is too short"); // @audit lenggh
```

Utils.sol#L262

In `SynthesisV2`, there is a typo on L161.

```
    * In cases when synth type is thisd party
    synth (ThirdPartySynth, ThirdPartyToken)
    - address must be a ISynthAdapter // @audit thisd
```

SynthesisV2.sol#L161

In `PoolAdapterAave`, there are typos on L88-89.

```
    * @param i Index of the input token
        (if > n, it's undrlying); // @audit undrlying
    * @param j Index of the output token
        (if > n, it's undrlying); // @audit undrlying
```

PoolAdapter.sol#L88-L89

Recommendation

We recommend correcting these typos.

Client's commentary

> fixed
> 3ea578b71fe5f1204bf81a503e0a332c76d4b675

L-8	Redundant `DifferentDecimalsAdapter.withdraw()` function to be removed
Severity	Low
Status	Fixed in 67f35d70

Description

The`SynthesisV2` contract always maintains the ownership of the `DifferentDecimalsAdapter` contract.
Within `DifferentDecimalsAdapter`, there exists a `withdraw()` function that is not callable from `Synthesis`, as a corresponding function is absent.

```
    function withdraw(uint256 amount) external onlyOwner {
        SafeERC20.safeTransfer(
            ISynthERC20(synthToken), owner(), amount
        );
    }
```

DifferentDecimalsAdapter.sol#L59-L61

Recommendation

We recommend removing the `withdraw()` function.

Client's commentary

> fixed
> b22c8faeb1389ff795d7e047e931d5eb813cdd5e

L-9	The `nonReentrant` modifier should occur `before` all other modifiers
Severity	Low
Status	Fixed in 67f35d70

Description

Reentrancy could occur in modifiers that are placed before `nonReentrant`. Several functions within the scope have this issue:

RouterV2.sol#L71

RouterV2.sol#L80

FeesTreasury.sol#L17

BalancerTreasury.sol#L32

Recommendation

We recommend placing `nonReentrant` before other modifiers.

Client's commentary

> fixed
> b98cfdb75813f41001331a00052aac37ca3598bd

L-10	Slippage value doesn't account for `bridgeFee`
Severity	Low
Status	Acknowledged

Description

The `bridgeFee` is applied each time the `unlock()` function in `PortalV2` and the `mint()` function in `SynthesisV2` are used.

PortalV2.sol#L89-L92

SynthesisV2.sol#L93-L96

The `minAmountOut` parameter is used during `ADD_CODE`, `REMOVE_CODE` and `SWAP_CODE` operations, and it doesn't account for `bridgeFee`. It is applied after these operations are completed, potentially resulting in users receiving `fewer` tokens than expected.

For example, a user calls the `ADD_CODE` operation with `10_000` USDT tokens and receives `9_900` lp tokens (slippage value is 2%), then the user locks `9_900` lp tokens and mints `9_800` synthLp tokens (`bridgeFee` value is `100`). Consequently, despite the expectation of `9_900` tokens, an extra `100` tokens are transferred to the treasury.

Recommendation

We recommend incorporating the `bridgeFee` into the slippage value calculation and notify users about the extra fee on the front-end.

Client's commentary

> Front-end should notify user about comission.

L-11	`@TODO` flag in Utils library
Severity	Low
Status	Acknowledged

Description

A `TODO` flag is present in the Utils library.

Utils.sol#L257

Recommendation

We recommend removing `TODO` and completing the comments to the `compressMCPubKey()` function.

Client's commentary

> ignored

L-12	Incompatibility with fee-on-transfer or rebasing tokens
Severity	Low
Status	Acknowledged

Description

Some parts of the code operate under the assumption that the `safeTransfer()` and `safeTransferFrom()` functions will transfer an exact amount of tokens.

However, the actual amount of tokens received may be less for `fee-on-transfer` tokens. Moreover, the protocol doesn't accommodate rebasing tokens whose balances fluctuate over time.

These discrepancies can lead to transaction reverts or other unexpected behaviour.

Recommendation

We recommend implementing verification checks both before and after token transfers.

Client's commentary

> known issue, fee-on-transfer is not supported for now.

L-13	Vulnerability to withdrawal of stuck funds in certain contracts
Severity	Low
Status	Acknowledged

Description

There is a vulnerability in the router, bridge, and synthesis contracts that allows for the withdrawal of stuck funds using specifically crafted parameters.

Router

For example, a malicious actor could withdraw Ether from the router contract by initiating an UNWRAP operation with a fake `p.tokenIn` contract. Then, when `_unwrap()` is executed, the call to `IWETH9(p.tokenIn).withdraw(p.amountIn)` would fail to function as intended, and the hacker will receive the Ether:

```
function _unwrap
  ...
  IWETH9(p.tokenIn).withdraw(p.amountIn);
  (bool sent, ) = p.to.call{ value: p.amountIn }("");
```

RouterV2.sol#L223

It is worth noting that, at the time of writing this report, there are native tokens stuck on the routers (approximately $2500 in total):

0xE7Db62...db4A3bE4.

Bridge

Similarly, funds can be extracted from the Bridge contract by exploiting the BURN_MINT operation for a token that lacks an adapter. By stacking LOCK_MINT and BURN_MINT operations, the funds are first locked on chain A and then unlocked on chain B. After that, executing BURN_MINT on chain B, with the bridge's `msg.sender`, allows for the burning of tokens from the bridge due to the bridge address itself being returned by the `_checkMaskedParams()` function:

```
function _checkMaskedParams
  ...
  if (currentFrom != address(0)) {
    require(currentFrom == msg.sender, "Router: wrong sender");
    from = currentFrom;
  } else {
    from = prevMaskedParams.to;
  }
```

RouterV2.sol#L255

SynthesisV2

It is also possible to attempt to withdraw stuck funds from SynthesisV2 contracts by executing operations with non-matching `amountIn` values: `WRAP(amountIn=1 wei), BURN_UNLOCK(amountIn=100 ether)`.

Notes

- There are native tokens currently stuck on the routers: 0xE7Db62...db4A3bE4
- The bridge does not accumulate user transfers by design.
- The router and synthesis contracts may accumulate stuck tokens only due to user errors or suboptimal `amountIn` and `amountOut` values.
- The outlined exploits could be mitigated with backend filters, which would require an attacker to bypass these protections to successfully exploit the system.

Recommendation

We recommend writing an additional test case or enhancing the existing ones to ensure that the router, synthesis, and bridge contracts do not accumulate funds after all transactions. Notably, having such tests will prevent accidental errors in the future when expanding functionality, especially if there arises a need to store some transfers on any of the specified contracts.

Client's commentary

> working on tests

L-14	Emergency Handling for Cross-Chain Operations
Severity	Low
Status	Acknowledged

Description

In the code located at RouterV2.sol#L167, it is explicitly expected that the `EMERGENCY` function will be invoked for operations `LOCK_MINT_CODE`, `BURN_UNLOCK_CODE`, and `BURN_MINT_CODE`. However, it should also be expected for operations `EMERGENCY_UNLOCK_CODE` and `EMERGENCY_MINT_CODE` because these operations perform cross-chain transitions between networks as well.

Recommendation

Although a revert will occur in this case, it is recommended to validate the `abi.decode` process to prevent any accidental inclusion of a `CancelParams` structure. This will add an additional layer of safety and prevent potential misinterpretations or incorrect handling of the parameters during emergency operations.

Client's commentary

> Emergency operation can be repeated if needed. But it can be executed only once.

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