PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Singularity DAO (DynaRouter)

29 May 2025

paladinsec.co

info@paladinsec.co

Table of Contents

Table of Contents 2

Disclaimer 4

1 Overview 5

1.1 Summary 5

1.2 Contracts Assessed 6

1.3 Findings Summary 7

1.3.1 DynaRouterRegistry 8

1.3.2 MetaDynaRouter 8

1.3.3 DynaRouter 9

1.3.4 BaseDynaRouter 9

1.3.5 TokenWhitelistRegistry 9

1.3.6 BalancerV2WeightedPoolOracle 9

2 Findings 10

2.1 DynaRouterRegistry 10

2.1.1 Privileged Functions 10

2.1.2 Issues & Recommendations 11

2.2 MetaDynaRouter 14

2.2.1 Privileged Functions 14

2.2.2 Issues & Recommendations 15

2.3 DynaRouter 19

2.3.1 Privileged Functions 19

2.3.2 Issues & Recommendations 20

2.4 BaseDynaRouter 22

2.4.1 Privileged Functions 22

2.4.2 Issues & Recommendations 22

2.5 TokenWhitelistRegistry 23

2.5.1 Privileged Functions 23

2.5.2 Issues & Recommendations 23
2.6 BalancerV2WeightedPoolOracle 24
2.6.1 Privileged Functions 24
2.6.2 Issues & Recommendations 25

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains the right to re-use any and all knowledge and expertise gained during the audit process, including, but not limited to, vulnerabilities, bugs, or new attack vectors. Paladin is therefore allowed and expected to use this knowledge in subsequent audits and to inform any third party, who may or may not be our past or current clients, whose projects have similar vulnerabilities. Paladin is furthermore allowed to claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for Singularity DAO's DynaRouter contracts on the Arbitrum network. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	Singularity DAO
URL	https://singularitydao.ai/
Platform	Base
Language	Solidity
Preliminary	https://github.com/Singularity-DAO/DynaVaults/commit/a830d27ad7e370125ab1cfda389c755e0ab4239b
Resolution	https://github.com/Singularity-DAO/DynaVaults/commit/08dd423aac6fc0634b9de721d110017590a8fe3d

1.2 Contracts Assessed

Contract	Address	Live Match
BaseDynaRouter	Dependency	MATCH
DynaRouterRegistry	0x82BccFF6E13a5B6e4D56E33a2079b009255D40f9	UNMATCHED
DynaRouter	-	UNUSED
MetaDynaRouter	0xa12071800e9aEe44d33E2Fda0555508Ee777d422	MATCH
TokenWhitelistRegistry	0x1c318265f9E207522aB827580F73EFb0B70d03Ef	MATCH
BalancerV2WeightedPool Oracle	0x915BE93926056B7ea7cF03b075fbc1d1d77Bafb2	MATCH

DynaRouterRegistry: https://www.diffchecker.com/G94CM4id/

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change)	Failed
Governance	0	-	-	-	-
High	3	3	-	-	-
Medium	2	1	-	-	-
Low	4	3	-	-	1
Informational	9	7	1	1	-
Total	**18**	**14**	**1**	**1**	**1**

Classification of Issues

Severity	Description
Governance	Issues under this category are where the governance or owners of the protocol have certain privileges that users need to be aware of, some of which can result in the loss of user funds if the governance's private keys are lost or if they turn malicious, for example.
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 DynaRouterRegistry

ID	Severity	Summary	Status
01	MEDIUM	`setEnableRouter()` does not clear `enabledIndex` after disabling router	PARTIAL
02	LOW	No way to reset pair routes	FAILED
03	INFO	Emit events on state changing functions	✓ RESOLVED
04	INFO	Insufficient validation	PARTIAL

1.3.2 MetaDynaRouter

ID	Severity	Summary	Status
05	HIGH	Improper native value sent in swap	✓ RESOLVED
06	HIGH	`swapConnectors()` might get DOSed if hanging allowances ever occur in `MetaDynaRouter`	✓ RESOLVED
07	LOW	`swapConnectors()` calls could revert if destination address implements logic in its `receive` function	✓ RESOLVED
08	LOW	`swapConnectors()` might revert with some token implementations	✓ RESOLVED
09	INFO	Swaps could be made with any arbitrary contract	✓ RESOLVED
10	INFO	Gas optimizations	✓ RESOLVED
11	INFO	Insufficient validation	✓ RESOLVED
12	INFO	Emit events on state changing functions	✓ RESOLVED

1.3.3 DynaRouter

ID	Severity	Summary	Status
13	HIGH	Arbitrary from value allows attacker to steal user funds via frontrunning	✓ RESOLVED
14	INFO	Emit events on state changing functions	✓ RESOLVED
15	INFO	Inherited method potentially not implemented	✓ RESOLVED

1.3.4 BaseDynaRouter

No issues found.

1.3.5 TokenWhitelistRegistry

No issues found.

1.3.6 BalancerV2WeightedPoolOracle

ID	Severity	Summary	Status
16	MEDIUM	`tokenReferenceValue` can be influenced by an attacker to report an incorrect `referenceValue`	
17	LOW	Prices returned from oracle not validated	✓ RESOLVED
18	INFO	Check oracle data staleness	ACKNOWLEDGED

2 Findings

2.1 DynaRouterRegistry

`DynaRouterRegistry` keeps track of the routers used in the DynaRouter framework. The admin can set the native, multi and token routers, enable/disable routers and configure specific routes for a pair of tokens.

2.1.1 Privileged Functions

- `setAllRouters [onlyRole]`
- `addRouters [onlyRole]`
- `setNativeRouters [onlyRole]`
- `setTokenRouters [onlyRole]`
- `setMultiTokenRouters [onlyRole]`
- `setPairRouters [onlyRole]`
- `addPairRoute [onlyRole]`
- `addDirectionalPairRoute [onlyRole]`
- `setEnableRouter [onlyRole]`
- `grantRole [onlyRole]`
- `revokeRole [onlyRole]`

2.1.2 Issues & Recommendations

Issue #01	**`setEnableRouter()` does not clear `enabledIndex` after disabling router**
Severity	MEDIUM SEVERITY
Description	If the admin enables four routers via `setEnableRouter()` and then disables the second one (`router2`), `enabledIndex[router]` is not being cleared to 0 during the disable. The fourth router (`router4`) will get copied in the place of the second in `enabledRouters[routerIndex]` and the fourth element in the array will get popped. `enabledIndex[router2 address]` returns index 1 (which is erroneous since `router2` is no longer enabled) and `enabledIndex[router4 address]` will return index 3 which is erroneous since we moved the 4th router to the second position so the correct index would be 1. This would prevent the admin from disabling `router4` since code would revert on L141 due to out of bounds error.
Recommendation	Consider updating `enabledIndex` of both routers that change positions in the array during removal operation.

Resolution	PARTIALLY RESOLVED
	
	During the resolution round, the team has uncovered a new issue—the last router can not be removed because the router and `lastRouter` will be the same and the index is basically reset for the `lastRouter`:
	
	`enabledIndex[router] = 0;`
	`enabledIndex[lastRouter] = routerIndex;`
	
	This should check if it is the last router to remove so that `enabledIndex` is not set back to the old index.

Issue #02	No way to reset pair routes
Severity	LOW SEVERITY
Description	The `pairRoutes` state variable can be manipulated through `addPairRoute()` and `addDirectionalPairRoute()`. Both of those functions push to the array and keep making it bigger. However, there is no function to decrease or reset the `pairRoutes` array as there is for the other defined state array variables. This can be a problem in case the array gets too big or if it is necessary to remove an element from it.
Recommendation	Implement an additional function that directly sets `pairRoutes`.
Resolution	FAILED RESOLUTION `pairRoutes` and `pairRouters` are different and there is no setter function for `pairRoutes` as of now. A setter function is present only for `pairRouters`.

Issue #03	Emit events on state changing functions
Severity	INFORMATIONAL
Description	The following state changing functions do not emit events: - `setAllRouters()` - `addRouters()` It is considered best practice to emit events that mark changes in the state of the smart contract. It provides transparency to anyone observing how the contract state has changed and also allows querying by off-chain listeners when that is necessary.
Recommendation	Consider adding events to the functions.
Resolution	RESOLVED

Issue #04	Insufficient validation
Severity	INFORMATIONAL
Description	Within `setPairRouters()`, make sure to check that `tokenIn != tokenOut`. — Within `setEnableRouter()`, check that the router variables is not `address(0)`. — Within `getDefaultPairRouter()`, consider checking if the `pairRouters` array has any elements for the provided `tokenIn` and return `address(0)`. If not, try to access index 0 directly instead of reverting.
Recommendation	Consider adding the recommended validations.
Resolution	PARTIALLY RESOLVED Only the last part was resolved.

2.2 MetaDynaRouter

`MetaDynaRouter` enables users to use any type of Dyna router to perform swaps from one single contract. The swapConnectors() function performs swaps between multiple Dyna routers in a single transaction.

2.2.1 Privileged Functions

- `setRouterRegistry [onlyRole]`
- `grantRole [onlyRole]`
- `revokeRole [onlyRole]`

2.2.2 Issues & Recommendations

Issue #05	Improper native value sent in swap
Severity	HIGH SEVERITY
Description	During a swap within `swapConnectors()`, the value is sent as native token if `tokenIn` is `USE_NATIVE` by using `msg.value` as `amount`: `NativeDynaRouterAPI(connectors[i].router).swap{value: tokenIn == USE_NATIVE ? msg.value : 0}(....)` The issue is that the code assumes `USE_NATIVE` is always the first token. However if we have a swap where ERC20 is swapped for `USE_NATIVE` and then `USE_NATIVE` is used as `tokenIn` in the next swap, then the actual amount to be send should be `amountIn` which was received from the first ERC20 -> `USE_NATIVE` swap. Depending on the swap order, this would lead to a revert or would send the wrong amounts.
Recommendation	Consider using `amountIn` instead of `msg.value`.
Resolution	RESOLVED

Issue #06	`swapConnectors()` might get DOSed if hanging allowances ever occur in `MetaDynaRouter`
Severity	HIGH SEVERITY
Description	In `swapConnectors()`, there is a check for the current allowance of MetaDynaRouter towards a router and if it is not enough, `approve()` is called with the allowance that will be enough to make the swap. The issue is that `approve()` does not increase the allowance but overwrites it.
Recommendation	The correct implementation should set the allowance to `amountIn` instead of `amountIn - allowanceIn`.
Resolution	RESOLVED

Issue #07	**swapConnectors() calls could revert if destination address implements logic in its receive function**
Severity	LOW SEVERITY
Description	`swapConnectors()` uses `payable(to).transfer(amountIn)` to transfer native funds to the destination address. The `transfer` function forwards only 2300 gas to the call which is likely not to be enough if the destination contract implements any logic inside its receive function.
Recommendation	Consider replacing `transfer` with `call` for the transfer of native funds.
Resolution	RESOLVED

Issue #08	**swapConnectors() might revert with some token implementations**
Severity	LOW SEVERITY
Description	Usage of `approve()` instead of `forceApprove()` could revert since some token implementations revert if you try to change the allowance from non-zero value to another non-zero value. If there are hanging allowances, `swapConnectors()` might revert.
Recommendation	Consider replacing `approve()` with `forceApprove()`.
Resolution	RESOLVED

Issue #09	Swaps could be made with any arbitrary contract
Severity	INFORMATIONAL
Description	`MetaDynaRouter` and `DynaRouter` can make swap calls to any arbitrary address that could be outside of the Dyna ecosystem without them being registered in `DynaRouterRegistry`.
Recommendation	Consider verifying with `DynaRouterRegistry` if the called router addresses are part of the Dyna ecosystem to avoid unexpected behaviour.
Resolution	RESOLVED

Issue #10	Gas optimizations
Severity	INFORMATIONAL
Description	Throughout the MetaRouter contract, the router arrays are fetched in full from the router registry (`getPairRouters()`, `getNativeRouters()`, etc) only to access the first element. This is inefficient in terms of gas and could theoretically lead to DOS if the array grows very big (which is not very likely).
Recommendation	Consider accessing the element directly from the state variable since it is public like this: `address router = DynaRouterRegistryAPI(routerRegistry).nativeRouters(0)` This issue applies to all functions that load the different types of router arrays.
Resolution	RESOLVED

Issue #11	Insufficient validation
Severity	INFORMATIONAL
Description	Within `setRouterRegistry()`, check that the `registryAddress` variables is not `address(0)`. — Within `swapNative()`, check that `msg.value` is 0 when `tokenIn` is not `USE_NATIVE`. The same consideration applies to `swapConnectors()`. — Within `swapConnectors()`, also validate the `minAmountsOut` length in addition to the connectors and `swapData` array length check.
Recommendation	Consider adding the recommended validations.
Resolution	RESOLVED

Issue #12	Emit events on state changing functions
Severity	INFORMATIONAL
Description	The following state changing functions do not emit events: - `setRouterRegistry()` It is considered best practice to emit events that mark changes in the state of the smart contract. It provides transparency to anyone observing how the contract state has changed and also allows querying by off-chain listeners when that is necessary.
Recommendation	Consider adding events to the functions.
Resolution	RESOLVED

2.3 DynaRouter

`DynaRouter` uses the MetaRouter to get the default router and estimated conversion for a particular swap. It also includes a swap function that will call the user specified router.

2.3.1 Privileged Functions

- `setMetaRouter [onlyRole]`
- `grantRole [onlyRole]`
- `revokeRole [onlyRole]`

2.3.2 Issues & Recommendations

Issue #13	**Arbitrary from value allows attacker to steal user funds via frontrunning**
Severity	HIGH SEVERITY
Description	Throughout the codebase, swap functions can be called with arbitrary from addresses which allows an attacker to frontrun the swap call of a user who approved a router in a previous separate transaction and change the to address to himself, effectively stealing the tokens of the user.
Recommendation	To support this architecture and to fix the vulnerability described above we need to introduce slight changes in the swap functions of all routers. We can separate the routers into 2 categories. The first are routers that implement logic in their swap function to perform a swap with a certain DEX (example: `AerodromeDynaRouter`, `UniV2DynaRouter`, `UniV3DynaRouter`)—for this issue, we will refer to them as implementation routers. The second type of routers redirect swap calls (such as `DynaRouter` and `MetaDynaRouter`) to the implementation routers mentioned in the first category— we will refer to these as redirecting routers. Implementation routers could be called by redirecting routers and directly by users. They should check if the caller is not a redirecting router—in such cases, the `from` parameter of the `swap` function should be changed (overwritten) to `msg.sender`. If the caller is a redirecting router, the `from` parameter should be left unchanged. Redirecting routers are intended to be called only by users, thus the `from` parameter of the `swap` function should be changed to `msg.sender` or this parameter should be removed entirely. For good sanitation, the `swap` function of a redirecting router should revert if another redirecting router is the caller. We could add the redirecting routers to the router registry where all contracts can verify if the caller is such a router.
Resolution	RESOLVED

Issue #14	Emit events on state changing functions
Severity	INFORMATIONAL
Description	The following state changing functions do not emit events: - `setMetaRouter()` It is considered best practice to emit events that mark changes in the state of the smart contract. It provides transparency to anyone observing how the contract state has changed and also allows querying by offchain listeners when that is necessary.
Recommendation	Consider adding events to the function.
Resolution	RESOLVED

Issue #15	Inherited method potentially not implemented
Severity	INFORMATIONAL
Description	`DynaRouter` inherits from `BaseDynarouter` and the inherited `previewSwapRoute()` method has not been overridden, returning the default values defined in the base contract.
Recommendation	In case the above is not an intentional design choice, make sure to override the function with proper implementation
Resolution	RESOLVED

2.4 BaseDynaRouter

`BaseDynaRouter` is an abstract contract implements the base logic of the DynaRouter.

2.4.1 Privileged Functions

- `grantRole [onlyRole]`
- `revokeRole [onlyRole]`

2.4.2 Issues & Recommendations

No issues found.

2.5 TokenWhitelistRegistry

`TokenWhitelistRegistry` allows the admin to manage a token whitelist per user. This contract is called by BalancerV2DynaZapper and VaultDynaZapper during swaps.

2.5.1 Privileged Functions

- `setEnforceTokenWhitelist [onlyRole]`
- `whitelistUserForToken [onlyRole]`
- `grantRole [onlyRole]`
- `revokeRole [onlyRole]`

2.5.2 Issues & Recommendations

No issues found.

2.6 BalancerV2WeightedPoolOracle

`BalancerV2WeightedPoolOracle` calculates the price of a BPT token amount in a certain asset based on the oracle prices of each asset in the pool.

2.6.1 Privileged Functions

- `initialize [onlyRole]`
- `grantRole [onlyRole]`
- `revokeRole [onlyRole]`

2.6.2 Issues & Recommendations

Issue #16	**`tokenReferenceValue` can be influenced by an attacker to report an incorrect `referenceValue`**
Severity	MEDIUM SEVERITY
Description	`tokenReferenceValue()` takes the balances of the pool and then uses an oracle to calculate the TVL. An attacker can front-run the `tokenReferenceValue()` call and execute a big swap (with or without a flash loan) that can change the balance ratios drastically. Then with the new token balances, the TVL will be calculated and this can result in increased/decreased TVL value since the price of the tokens reported by the oracle will be unchanged and naturally some tokens could have a bigger/lower price than others.
Recommendation	Treat `BalancerV2WeightedPoolOracle.tokenReferenceValue()` as a spot price that can be easily influenced.
Resolution	

Issue #17	**Prices returned from oracle not validated**
Severity	LOW SEVERITY
Description	Within `_getPrice()`, make sure to validate that the returned `priceInQuote` is not 0 when calling the reference oracle to get the price for a token, as this means that the oracle has malfunctioned in some way.
Recommendation	Consider adding a check to ensure `priceInQuote !=0`.
Resolution	RESOLVED

Issue #18	Check oracle data staleness
Severity	INFORMATIONAL
Description	When getting prices from the reference asset oracle, make sure to compare the observation timestamp value to some staleness threshold to make sure the prices are up to date. Such validation has been implemented in `ReferenceAssetOracle` but it seems to be lacking in `ChainlinkOracle` which also implements the `IReferenceAssetOracle` interface. It is not clear which reference oracle will be configured, so it would be a good safety measure to implement the staleness check inside the Balancer oracle contract as well.
Recommendation	Check that the returned observation timestamps are not stale.
Resolution	ACKNOWLEDGED


PALADIN
BLOCKCHAIN SECURITY