Mantle Network (Token Migration)

Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

Type	Token Migration
Timeline	2023-06-01 through 2023-06-12
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	BitDAO Proposal (MIP-22) BitDAO Proposal (BIP-21)
	• mantlenetworkio/ma

Documentation quality	Medium
Test quality	Medium
Total Findings	6 Fixed: 3 Acknowledged: 2 Mitigated: 1
High severity findin	1 Fixed: 1

Source Code	ntle-token-contracts #b2016df
Auditors	• Guillermo Escobero Auditing Engineer • Julio Aguliar Auditing Engineer • Mustafa Hasan Senior Auditing Engineer • Roman Rohleder Senior Auditing Engineer

gs	
Medium severity findings	1 Acknowledged: 1
Low severity findings	3 Fixed: 1 Acknowledged: 1 Mitigated: 1
Undetermineds	0

Severity findings	
Informational findings	1 Fixed: 1

Summary of Findings

Quantstamp performed an audit for the token migration plan of Mantle Network of migrating the `$BIT` token to the `$MNT` token based on the code present in the listed repositories.

BitDAO community decided in BIP-21 and MIP-22 proposals to start a token migration plan to optimize and unify the Mantle ecosystem. Through a smart contract, users will be allowed to convert `$BIT` tokens to `$MNT` tokens. This operation can only be performed in one direction, depositing `$BIT` to obtain `$MNT` and not the other way around. MNT-1 issue covers this and tries to bring the Mantle team's attention to confirm the token conversion ratio.

The token contract owner is responsible for minting and transferring the new `$MNT` tokens to the migration contract. Owner functions are limited, he can only mint once every year and is limited by an inflation cap (modifiable, but capped to 20%). However, we recommend protecting this privileged wallet with a multi-signature

mechanism to avoid possible minting to an arbitrary address. We recommend being extra careful when transferring the ownership of this contract as well, as future mints will not be possible if the new owner's address is not controlled by Mantle.

Regarding code coverage, the repository shows an average of 75% branch coverage. We recommend increasing it to at least 90%.

Several issues were found in the auditing process. All of them are discussed in this report, including some best practices and documentation recommendations. We recommend addressing all of them.

After fix-review: The developers addressed all issues by either fixing or acknowledging them. Issue MNT-6 (Old Solidity Version) was mitigated, as the solidity version was upgraded to `0.8.15`, while we recommend upgrading to `0.8.18`. We still recommend raising the code coverage metrics.

ID	DESCRIPTION	SEVERITY	STATUS
MNT-1	Adherence To Specification: `BIT` Holders Might Get Less `MNT`	• High	Fixed
MNT-2	Defund Protection Bypassed	• Medium	Acknowledged
MNT-3	Missing Input Validation	• Low	Fixed
MNT-4	Unnecessary `receive()` and `fallback()` Functionality	• Informational	Fixed
MNT-5	Ownership Can Be Renounced	• Low	Acknowledged
MNT-6	Old Solidity	• Low	Mitigated

ID	DESCRIPTION	SEVERITY	STATUS
	Version		

Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

> **Disclaimer**
> Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
    1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
    2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:

1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Scope

Files Included

- `L1MantleToken.sol`
- `MantleTokenMigrator.sol`

Findings

MNT-1

Adherence To Specification: `BIT` Holders Might Get Less `MNT`

• High Fixed

Update

Fixed in commit b187163 by removing the numerator and denominator functionality and minting `MNT` tokens 1:1 to input `BIT` tokens, as described in the specification.

Update

Marked as "Fixed" by the client. Addressed in: `b18716323f4054120856247ec5e3c28aa3bdb5da` .

Update

Marked as "Fixed" by the client. Addressed in: `b18716323f4054120856247ec5e3c28aa3bdb5da` .

File(s) affected: `MantleTokenMigrator.sol`

Description: The proposal in BitDAO states:

> "The one-way token conversion ratio shall be set at 1 $BIT token to 1 $MNT token".

However, the `MantleTokenMigrator` contract computes the ratio based on `TOKEN_CONVERSION_NUMERATOR` and `TOKEN_CONVERSION_DENOMINATOR`, possibly not guaranteeing the 1:1 ratio.

Recommendation: We recommend removing such constants and conversion calculations or updating the proposal and other related documentation.

MNT-2
Defund Protection Bypassed

Medium | Acknowledged

Update

Acknowledged by the developers with:

> The owner of the contract will be set to a multisig after deployment, we added in a comment to clarify this.

Further, they added a corresponding comment in commit 51ea647.

Update

Marked as "Acknowledged" by the client. Addressed in: `51ea6473b44cae229589783da4e0eda29f57d40e`. The client provided the following explanation:

> The owner of the contract will be set to a multisig after deployment, we added in a comment to clarify this.

Update

Marked as "Acknowledged" by the client. Addressed in: `51ea6473b44cae229589783da4e0eda29f57d40e`. The client provided the following explanation:

> The owner of the contract will be set to a multisig after deployment, we added in a comment to clarify this.

File(s) affected: `MantleTokenMigrator.sol`

Description: `defundContract()` can be used by the owner of the contract to transfer the funds from the migrator contract to the `treasury` address. However, the owner can just change the `treasury` variable with the same privileges and therefore transfer the funds to an arbitrary address.

Recommendation: Confirm this is the expected behavior. If not, `defundContract()` and `setTreasury()` should be protected by different roles or keys (a multi-signature wallet approach would mitigate this as well). In this case, a malicious owner will only be able to transfer the funds to a fixed treasury address by another guardian or administrator.

MNT-3 Missing Input Validation

• Low ⓘ Fixed

Update

Fixed in commit 36c01ea.

1. Deprecated, as numerator/denominator functionality was removed.
2. Fixed.

Update

Marked as "Fixed" by the client. Addressed in: `36c01ea5eaf2d01415291be1d2d02c175b0e52ce` .

Update

Marked as "Fixed" by the client. Addressed in: `36c01ea5eaf2d01415291be1d2d02c175b0e52ce` .

File(s) affected: `MantleTokenMigrator.sol`

Related Issue(s): SWC-123

Description: It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. The following is a non-exhaustive list of missing input validations:

1. `MantleTokenMigrator.constructor()` : the `TOKEN_CONVERSION_NUMERATOR` has to be less than or equal to `TOKEN_CONVERSION_DENOMINATOR` . Additionally, the decimals of `BIT` and `MNT` tokens should be equal according to the NatSpec comments.
2. `MantleTokenMigrator.setTreasury()` needs a check against `address(0)` . Otherwise, the `defundContract()` , might just burn tokens.

Recommendation: We recommend adding the relevant checks.

MNT-4 Unnecessary `receive()` and `fallback()` Functionality

Informational | Fixed

> **Update**
> Fixed in commit 0a16e7c by removing the `receive()` and `fallback()` functionality, as suggested.

> **Update**
> Marked as "Fixed" by the client. Addressed in: `0a16e7cdbd251e850ff5514bf04e48ee9065848d` .

> **Update**
> Marked as "Fixed" by the client. Addressed in: `0a16e7cdbd251e850ff5514bf04e48ee9065848d` .

File(s) affected: `MantleTokenMigrator.sol`

Description: `MantleTokenMigrator` contract implements `receive()` and `fallback()` functions. Based on the documentation and project use cases, the contract does not need to receive ETH at any point. The implementation of the functions also shows the Mantle team's intention: to make the transaction revert when receiving ETH in this contract.

Recommendation: Remove `receive()` and `fallback()` functions from `MantleTokenMigrator` as the contract is not expected to receive ETH. This will make the contract more clear.

MNT-5 Ownership Can Be Renounced

Low | Acknowledged

> **Update**
> Acknowledged by the developers with:
>
> > For L1MantleToken, renounceOwnership is not an expected behavior and will not be instantiated by the governance

multisig. To prevent transferring ownership to a new owner address which is wrong or not controlled by the team we can implement the following measures:

We already have a permanent multisig address and plan to transfer ownership to that address after deploying the contract (for both contracts).

After transferring ownership to the multisig address, the owner will not instantiate a transfer to another owner.

Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

> For L1MantleToken, renounceOwnership is not an expected behavior and will not be instantiated by the governance multisig. To prevent transferring ownership to a new owner address which is wrong or not controlled by the team we can implement the following measures:
>
> - We already have a permanent multisig address and plan to transfer ownership to that address after deploying the contract (for both contracts).
> - After transferring ownership to the multisig address, the owner will not instanciate a transfer to another owner.

Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

> For L1MantleToken, renounceOwnership is not an expected behavior and will not be instantiated by the governance multisig. To prevent transferring ownership to a new owner address which is wrong or not controlled by the team we can implement the following measures:
>
> - We already have a permanent multisig address and plan to transfer ownership to that address after deploying the contract (for both contracts).
> - After transferring ownership to the multisig address, the owner will not instanciate a transfer to another owner.

File(s) affected: `L1MantleToken.sol` , `MantleTokenMigrator.sol`

Description: If the owner renounces their ownership, or the ownership is transferred to the wrong address, all ownable contracts will be left without an owner. During the audit, the auditing team found the following issues:

1. `L1MantleToken` :
    - `OwnableUpgradeable` from OpenZeppelin implements `renounceOwnership()` function. If the contract owner calls this function, all the functions guarded by `onlyOwner` will no longer be able to be executed.
    - `OwnableUpgradeable` implements `transferOwnership()` . If the new owner address is wrong or not controlled by the team, access to privileged functions (`onlyOwner`) will be lost forever.
2. `MantleTokenMigrator` :
    - This contract implements a custom function for transferring the contract ownership: `transferOwnership()` . If the new `owner` address is wrong or not controlled by the team, access to privileged functions (`onlyOwner`) will be lost forever.

Recommendation:

1. `L1MantleToken` :
    - Confirm that this is the intended behavior. If not, override and disable the `renounceOwnership()` function.
    - Consider using a two-step process when transferring the ownership of the contract (e.g. `Ownable2StepUpgradeable` from OpenZeppelin).
2. `MantleTokenMigrator` :
    - Consider using a two-step process when transferring the ownership of the contract (e.g. `Ownable2Step` from OpenZeppelin).

MNT-6 Old Solidity Version

• Low Mitigated

Update

Mitigated in commit 313ed11 by upgrading to solidity version `0.8.15` for both contracts.

Update

Marked as "Fixed" by the client. Addressed in: `313ed116b407360b100496d366212df77c341c1a` . The client provided the following explanation:

> We prefer to use an older more battle-tested compiler version (versus greater efficiency) for security reasoning. Updating to 0.8.15

> **Update**
> Marked as "Fixed" by the client. Addressed in: `313ed116b407360b100496d366212df77c341c1a` . The client provided the following explanation:
>
> > We prefer to use an older more battle-tested compiler version (versus greater efficiency) for security reasoning. Updating to 0.8.15

File(s) affected: `L1MantleToken.sol` , `MantleTokenMigrator.sol`

Description: The codebase is compiled using the `0.8.13` version of Solidity. Several versions have been released after that, fixing bugs and improving optimization.

Recommendation: We recommend using a newer compiler version. Declare `pragma solidity 0.8.18` in the contracts.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2)

business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Code Documentation

1. Each repository's `README.md` file should contain basic instructions on how to run the test suite and any prerequisites for running tests. For each of these prerequisites also specify the versions supported/tested.
2. Typographical errors: (Update: **Unresolved**)
    - `L1MantleToken.sol#L117` : `MintCapNumeratorSet` → `MintCapNumeratorChanged` .
    - `MantleTokenMigrator.sol#L220` : `tokens, in` → `tokens in` .
    - `MantleTokenMigrator.sol#L305` : `must not the` → `must not be the` .
    - `MantleTokenMigrator.sol#L311` : `addres` → `address` .

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

- `440...de5 ./contracts/Migration/MantleTokenMigrator.sol`
- `7e7...674 ./contracts/L1/L1MantleToken.sol`

Tests

- `9a5...b43 ./test/L1MantleToken.test.ts`
- `1f7...07a ./test/foundry/L1MantleToken.t.sol`
- `93f...c06 ./test/foundry/MantleTokenMigrator.t.sol`

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Slither v0.8.3

Steps taken to run the tools:

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Automated Analysis

Slither

Slither was used to get a static analysis of the repository. All the issues and recommendations are discussed in this report or classified as false positives.

Test Suite Results

All tests passed. Two test suites exist in the project, using hardhat and foundry libraries.

```
Running 11 tests for
test/foundry/MantleTokenMigrator.t.sol:MantleTokenMigratorT
est
[PASS] test_TreasuryCannotBeSetToZeroValues() (gas: 61514)
[PASS] test_defundContract() (gas: 220374)
[PASS] test_fallback() (gas: 23568)
[PASS] test_haltContract() (gas: 18951)
[PASS]
test_mantleTokenMigratorCannotBeInitializedWithZeroValues()
(gas: 142599)
[PASS] test_mantleTokenMigratorCorrectlyInitialized() (gas:
79052)
[PASS] test_migrateAllBIT() (gas: 173379)
[PASS] test_migrateBIT() (gas: 211621)
[PASS] test_setTreasury() (gas: 97537)
[PASS] test_sweepTokensERC20() (gas: 122743)
[PASS] test_transferOwnership() (gas: 99314)
Test result: ok. 11 passed; 0 failed; finished in 14.83ms

Running 15 tests for
test/foundry/L1MantleToken.t.sol:L1MantleTokenTest
[PASS] testMintTooEarly4() (gas: 189246)
[PASS] testMintTooMuch1() (gas: 99349)
[PASS] testMintTooMuch2() (gas: 144771)
```

```
[PASS] test_Info() (gas: 33868)
[PASS] test_Initialize() (gas: 20537)
[PASS] test_Mint() (gas: 98376)
[PASS] test_MintFuzz(uint256,uint256) (runs: 256, μ:
102539, ~: 105229)
[PASS] test_MintOnlyOwner() (gas: 20451)
[PASS] test_MintTooEarly1() (gas: 118374)
[PASS] test_MintTooEarly2() (gas: 143379)
[PASS] test_MintTooEarly3() (gas: 143754)
[PASS] test_MintTooEarlyFuzz(uint256,uint256,bool) (runs:
256, μ: 141507, ~: 149291)
[PASS] test_MintTooMuchFuzz(uint256,uint256) (runs: 256, μ:
148074, ~: 148930)
[PASS] test_setMintCapNumeratorFuzz(uint256) (runs: 256, μ:
39654, ~: 43230)
[PASS] test_setMintCapNumeratorTooLargeFuzz(uint256) (runs:
256, μ: 117980, ~: 118447)
Test result: ok. 15 passed; 0 failed; finished in 232.86ms

$ npx hardhat test

  L1MantleToken
    Upgrades
✔ 0xD2547e4AA4f5a2b0a512BFd45C9E3360FeEa48Df (transparent)
proxy ownership transfered through admin proxy
      ✓ Should transfer success when use
transferProxyAdminOwnership
    Info
      ✓ Should get some token infos
    Mint
      ✓ Should mint some tokens
      ✓ Should get mint owner
      ✓ Should mint when mint the rules
      ✓ Should fail when mint doesn't meet the rules
    Transfer
      ✓ Should transfer tokens between users
      ✓ Should fail to transfer with low balance
    Vote
      call
        set delegation
          ✓ delegation with balance
          ✓ delegation without balance

  10 passing (2s)
```

Code Coverage

File	% Lines	% Statements	% Branches	% Funcs
contracts/L1/L1MantleToken.sol	54.55% (**12/**22)	56.00% (**14/**25)	75.00% (**6/**8)	66.67% (**4/**6)
contracts/Migration/MantleTokenMigrator.sol	92.00% (**23/**25)	88.89% (**24/**27)	87.50% (**7/**8)	88.89% (**8/**9)
contracts/Mock/MockERC20.sol	0.00% (**0/**1)	0.00% (**0/**1)	100.00% (**0/**0)	0.00% (**0/**1)
test/foundry/mocks/EmptyContract.sol	0.00% (**0/**1)	0.00% (**0/**1)	100.00% (**0/**0)	0.00% (**0/**1)
test/foundry/mocks/LiquidStakingToken.sol	0.00% (**0/**5)	0.00% (**0/**5)	0.00% (**0/**2)	0.00% (**0/**3)
test/foundry/mocks/Multiplication.sol	0.00% (**0/**6)	0.00% (**0/**5)	0.00% (**0/**1)	0.00% (**0/**1)
test/foundry/mocks/ServiceManagerMock.sol	0.00% (**0/**2)	0.00% (**0/**2)	100.00% (**0/**0)	0.00% (**0/**5)
test/foundry/utils/UserFactory.sol	0.00% (**0/**9)	0.00% (**0/**14)	100.00% (**0/**0)	0.00% (**0/**2)

File	% Lines	% Statements	% Branches	% Funcs
Total	49.30% (**35/**71)	47.50% (**38/**80)	68.42% (**13/**19)	42.86% (**12/**28)

Changelog

- 2023-06-12 - Initial report
- 2023-06-16 - Final report

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication or other making available of the report to you by Quantstamp.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites&aspo; owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that your access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other

programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

 Mantle Network (Token Migration)