PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Material Network

16 January 2022

paladinsec.co

info@paladinsec.co

Table of Contents

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 full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Paladin is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. Paladin may, at its discretion, claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for Material Network on the Ethereum 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	Material Network
URL	https://www.material.network/
Platform	Ethereum
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
MTRL	0x13c99770694f07279607a6274f28a28c33086424	✓ MATCH
MTRLVesting	0x8071Db05f6f3D78C31b2a157348D866B4B9339fe	✓ MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
🔴 High	2	2	-	-
🟠 Medium	2	1	-	1
🟡 Low	4	2	-	2
🟣 Informational	2	1	-	1
Total	**10**	**6**	**-**	**4**

Classification of Issues

Severity	Description
🔴 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 MTRL

ID	Severity	Summary	Status
01	HIGH	Admin can disable all token transfers without any delay	✓ RESOLVED
02	LOW	Inconsistency between usage of `_msgSender()` and `msg.sender`	ACKNOWLEDGED
03	LOW	Lack of event emission for sensitive state changes	ACKNOWLEDGED

1.3.2 MTRLVesting

ID	Severity	Summary	Status
04	HIGH	Divide by zero due to lack of non zero check for `UNLOCK_CYCLE` will result in claim always reverting	✓ RESOLVED
05	MEDIUM	Compromise of admin private key could result in tokens being vested to a different wallet	ACKNOWLEDGED
06	MEDIUM	Lack of maximum check for `vestingStartBlock`	✓ RESOLVED
07	LOW	Tokens will not be released for an index if that index passes and claim is not called	✓ RESOLVED
08	LOW	Tokens can be claimed for index 0	✓ RESOLVED
09	INFO	Beneficiary can be a smart contract which can tokenize timelocked tokens for selling	ACKNOWLEDGED
10	INFO	`vestingStartBlock` can be before deployment block height	✓ RESOLVED

2 Findings

2.1 MTRL

The MTRL token is an ERC20 token which is extended with EIP-2612 permit capabilities. When deployed, 100,000,000 tokens will be minted to the admin address. No further minting of tokens can be done after that.

The admin has the privilege to enable and disable all token transfers using the `setTransfersAllowed` function.

2.1.1 Token Overview

Address	`0x13c99770694f07279607a6274f28a28c33086424`
Token Supply	100,000,000
Decimal Places	18
Transfer Max Size	No maximum
Transfer Min Size	No minimum
Transfer Fees	None

2.1.2 Privileged Roles

The following functions can be called by the admin of the contract:

- `setTransfersAllowed`
- `transferOwnership`

2.1.3 Issues & Recommendations

Issue #01	**Admin can disable all token transfers without any delay**
Severity	● HIGH SEVERITY
Description	The `setTransfersAllowed` allows the admin to disable all token transfers by setting `transfersAllowed` to false.
Recommendation	Consider if there really is a need to allow a privileged role to disable all token transfers. If this is required, consider putting this function behind a timelock, and making the privileged address a multi-sig solution. This is to prevent the compromise of the private key of the privileged address resulting in all transfers being disallowed.
Resolution	✓ RESOLVED The team has transferred the admin role to a burn address: https://etherscan.io/tx/0x962689e30ff0b8d6f04da4f61439ac28fda5f746f641428e1ad0e42c8c2e2f56

Issue #02	**Inconsistency between usage of `_msgSender()` and `msg.sender`**
Severity	● LOW SEVERITY
Description	There are instances in the code where `msg.sender` is used, and others where `_msgSender()` is used. This inconsistency could cause some issues as `_msgSender()` returns `msg.sender` for regular transactions, but for meta transactions, it can be used to return the end user instead of the relayer.
Recommendation	Consider using `_msgSender()` to replace all instances of `msg.sender`. Also consider using the OpenZeppelin Ownable contract to replace the admin with owner.
Resolution	● ACKNOWLEDGED

Issue #03	Lack of event emission for sensitive state changes
Severity	● LOW SEVERITY
Description	The `transferOwnership` transfers the admin role to a new address from the current admin, but does not emit an event.
Recommendation	Consider using the OpenZeppelin Ownable contract to replace the admin with owner.
Resolution	● ACKNOWLEDGED

2.2 MTRLVesting

The vesting wallet contract allows anyone to lock the ERC20 token specified at deployment in the constructor for a beneficiary address for a period of time. For every cycle of unlock cycle of blocks that have passed, anyone can call the `claim` function which will release 1,000,000 tokens to the beneficiary address.

There is a privileged admin role which can modify the beneficiary address.

2.2.1 Privileged Roles

The following functions can be called by the admin:

- `setWallet`
- `transferOwnership`

2.2.2 Issues & Recommendations

Issue #04	**Divide by zero due to lack of non zero check for UNLOCK_CYCLE will result in claim always reverting**
Severity	HIGH SEVERITY
Location	Line 85 `index = (block.number - vestingStartBlock) / UNLOCK_CYCLE;`
Description	As there is no check in the constructor for the value of UNLOCK_CYCLE, if set to zero, it will cause this line to revert by a divide by zero.
Recommendation	Consider adding a maximum and minimum value check for UNLOCK_CYCLE in the constructor. This will not only prevent the value from being zero, but also prevent it from being too high, resulting in a lock that would take a very long time to claim.
Resolution	✓ RESOLVED A maximum and minimum value check for UNLOCK_CYCLE has been added in the constructor.

Issue #05	**Compromise of admin private key could result in tokens being vested to a different wallet**
Severity	MEDIUM SEVERITY
Description	As the admin has the privilege to change the wallet address which claims unlocked tokens, if compromised, an attacker could change the wallet address to receive the vesting, and then change the admin address to prevent the wallet address from being changed to the correct address.
Recommendation	Consider making the privileged address a multi-sig solution. This is to prevent the compromise of the private key of the privileged address resulting in all transfers being disallowed.
Resolution	ACKNOWLEDGED

Issue #06	Lack of maximum check for `vestingStartBlock`
Severity	● MEDIUM SEVERITY
Description	As there is a lack of maximum check for `vestingStartBlock`, a vesting start block which is very far in the future could be set (for example, due to human error). If funds have been transferred to the contract, those funds could be stuck permanently.
Recommendation	Add a maximum check from the block number of the contract deployed to ensure that the `vestingStartBlock` is set to a reasonable time.
Resolution	✓ RESOLVED `vestingStartBlock` is now 1 block after the contract is deployed.

Issue #07	Tokens will not be released for an index if that index passes and `claim` is not called
Severity	● LOW SEVERITY
Description	If index 1 is claimed, but for the period of index 2, no claim is done, if the next time claim is called results in the index being 3, index 2's token amount would be skipped and not released.
Recommendation	Consider adding a check from the last claimed index to the current index to claim, allowing unclaimed indexes in between to also be claimed.
Resolution	✓ RESOLVED The `claim` function now will attempt to claim from the last claimed index until the current index, if there are enough tokens to be claimed.

Issue #08	Tokens can be claimed for index 0
Severity	● LOW SEVERITY
Description	Tokens can be claimed for the index starting at 0 instead of 1. This allows claiming immediately once the vestingStartBlock has passed, instead of having to wait at least for the UNLOCK_CYCLE to pass at least once.
Recommendation	Confirm if tokens should start claiming from index 0 or index 1. If it is the latter, add a check in claim to ensure that the index is greater than 0.

Resolution	✓ RESOLVED
	The following check has been added to ensure that claiming can only start after the `UNLOCK_CYCLE` has at least been elapsed once:
	`uint256 passedBlocks = block.number - vestingStartBlock;`
	`require(passedBlocks >= UNLOCK_CYCLE, 'claim: not claimable yet');`

Issue #09	Beneficiary can be a smart contract which can tokenize timelocked tokens for selling
Severity	● INFORMATIONAL
Description	As mentioned in the following blog post, it would be possible for a beneficiary to trustlessly sell their timelocked tokens without waiting for the timelock to expire. https://blog.openzeppelin.com/bypassing-smart-contract-timelocks/
Recommendation	It is recommended for the beneficiary to sign a non-arbitrary message and to check that the address derived from the signature using `ecrecover` tallies with the beneficiary address.
Resolution	● ACKNOWLEDGED

Issue #10	vestingStartBlock can be before deployment block height
Severity	● INFORMATIONAL
Description	During contract deployment, there is no check to ensure that the `vestingStartBlock` is greater than or equal to the block height of deployment.
Recommendation	Confirm if the `vestingStartBlock` should be at least greater than or equal to the block height at the time of deployment.
Resolution	✓ RESOLVED `vestingStartBlock` is now 1 block after the contract is deployed.

PALADIN
BLOCKCHAIN SECURITY