SHIELD NETWORK

Meta Arcade Token Audit

Audit Report SHIELD-METARC-1

Revision 1

Shield Network Auditing

March 10, 2022

Shieldnetwork.io

audits@shieldnetwork.io

Contents

1 Legal Disclaimer

Shield Network Auditing provides this report for informational purposes only. Shield Network is not liable under any circumstance for any third party reliance on this report or any information included herein. This report is not to be distributed, referenced, or otherwise utilized by any person or third party for any purpose without Shield Network's express prior written permission. Use of and access to this report and the information in the report is subject to these terms of use and all applicable laws.

This report is provided "as is" and without any warranties of any kind. You bear all risks associated with the use of this report and content, including without limitation, any reliance on the accuracy, completeness or usefulness of any content available in the report. Shield Network and its members and affiliates disclaim all warranties, express or implied, including without limitation, any warranties of title, non-infringement, accuracy, completeness, usefulness, merchantability, and fitness for a particular use, and all warranties arising from course of dealing, usage, or trade practice.

This report is for informational purposes only and should not be misconstrued as investment, financial, or other advice, nor should it be considered an endorsement or disapproval of any particular project or team or any mentioned third parties. Shield Network does not provide any guarantee or warranty regarding the security and functionality of the analyzed technology and does not claim the technology is bug-free.

This report is provided on the date documented on the title page and in the Overview section. Shield Network does not take any obligation or responsibility to provide further reports or updates on the audited project regardless of any emerging factors or changes, nor does it take any obligation to inform of any such changes that affect validity of this report in any way.

This Shield Network Auditing report may contain or reference links to third party websites. These links are being provided as a convenience and for informational purposes only; they do not constitute an endorsement or an approval by Shield Network of any of the content, products, or services contained in any third party website. Access to any third party website is at your own risk and you acknowledge and understand that linked third party websites may contain separate terms and privacy policies differing from Shield Network and this report. Shield Network bears no responsibility for the accuracy, legality or content of any third party website nor has any liability for the privacy or other practices of any such third party.

2 Introduction

This is the official Shield Network Auditing audit report for the Meta Arcade token contract. The scope of this audit is the `MetaArcade` contract and directly inherited contracts. External contracts and libraries such as Pancakeswap were taken into account but not audited directly. An initial report document was provided to the Meta Arcade team and this final audit contains all findings with resolutions and responses from the team.

2.1 Project Overview

Project Name	Meta Arcade
Website	`https://meta-arcade.net/`
Description	Meta Arcade is a P2E game that aims to be a multi chain token and NFT provider. We will raise funds from nft sales as well as tokens on BSC, Fantom, Avalanche and Solana. The game is playable on multiple platforms including the Google Play and Windows Store.
Platform	Binance Smart Chain (as audited)
Language	Solidity (v0.8.11)
Audited Contract	0xC737742Bb7e246227Aa70FC9469f8b26B900fE1D
Final Contract	0xdf41c31bbe17fafab4b5a172faaa5755aac772f0
Audit Conducted by	0xherman — Shield Network Auditing
Report Date	March 10, 2022

2.2 Risk Classifications

Severity	Definition
Critical	Critical risk level should be fixed immediately. These findings cause severe failure in desired functionality or can be exploited easily leading to loss and manipulation of funds and data.
Major	Major risk level findings are very important to fix. These findings detail vulnerabilities that are hard or privileged to exploit but can lead to critical risk level issues of loss and manipulation of funds and data.
Medium	Medium risk level finding fixes are highly recommended. These findings may or may not directly impact the desired functionality but carry a risk of future exploits or manipulation.
Minor	Minor risk level findings fixes are recommended. These findings will likely not cause any impact to desired contract functionality, but could lead to unexpected issues in the future.
Informational	Informational level findings can be safely ignored. These generally fall into code style recommendations, minor optimizations, or basic discussions and do not affect desired functionality of contract.

2.3 Audit Methodology

Manual Code Review, Testnet Deployment, Automated Tests

When performing an audit, Shield Network Auditing utilizes three main methodologies to create an initial report for the project team. The team then has a chance to resolve, partially resolve, or acknowledge all findings in the initial report. After continuing use of audit methodologies on any contract changes returned by the project team, a final audit report is published by Shield Network Auditing. Audit methodologies used for this report generally fall into three main categories:

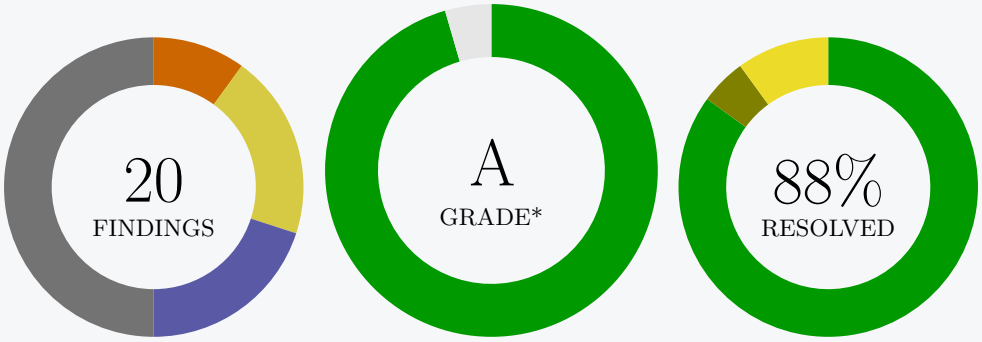
1. **Manual code review**
   During this process, the contract code is reviewed line-by-line to understand desired smart contract functionality and check for any visible vulnerabilities or bugs per auditor knowledge or a common issue checklist. This process is often repeated over multiple sessions and generally produces the majority of audit findings. These findings are then verified through other processes. Manual code review is always active and integrated into all other processes.

2. **Testnet deployment**
   For this process, the smart contract code under audit is deployed to a testnet or test environment which best mimics the current or expected future mainnet environment. After deployment to testnet, manual tests are run first targeting all at risk functionality and then performed ad hoc to ensure all general functionality performs as expected.

3. **Automated tests**
   If any automated tests (unit or integration tests) are provided by the project under audit, all tests will be run to ensure they pass and reviewed to ensure they accurately cover and test desired functionality. If no automated tests exist or tests do not adequately cover functionality, new tests are written under this process to better test the contract. Automated tests are also written to prove findings (if possible with an automated test), and to ensure any changes made by the project team fully resolve the issue.

These methodologies are the bare minimum performed in any audit by Shield Network Auditing and do not necessarily constitute the full audit performance. If the project team decides to make changes to the contract to resolve findings from an initial audit report, all methodologies are performed again on the updated codebase.

3 Audit Findings



*Grade is not an endorsement or disapproval and is automatically calculated based on unresolved findings and their severity levels

Severity	Found	Resolved	Partially Resolved	Acknowledged
Critical	0	0	0	0
Major	2	2	0	0
Medium	4	3	1	0
Minor	4	4	0	0
Informational	10	8	0	2
Total	20	17	1	2

3.1 Critical Findings (0)

None

3.2 Major Findings (2)

Issue #1	No Maximum Fee
Severity	Major Resolved
Description	The owner can set various fees to any value, including a total tax greater than 100% of transaction amount which would revert due to a subtraction overflow. This would halt trading except for addresses the owner excludes from fees.

Location	`setLiquidityFeePercent`
	`setMarketingFeePercent`
	`setDevFeePercent`
	`setGameRewardsFeePercent`

Recommendation	Recommended to require fees be less than a maximum value (e.g. 10%).
Resolution	Resolved by limiting each fee to maximum value of 10% and total fees to a maximum of 30%.

Issue #2	No Minimum Maximum Transaction
Severity	Major Resolved
Description	The owner can set maximum transaction to any value including 0. This would halt trading except for addresses the owner whitelists to avoid maximum transaction.
Location	`setMaxTxPercent` `setMaxTxAmount`
Recommendation	Recommended to require maximum transaction be greater than a minimum value.
Resolution	Resolved by limiting maximum transaction to minimum value of 0.01% of total supply.

3.3 Medium Findings (4)

Issue #3	No Tax Transfer Events
Severity	Medium Resolved
Description	No `Transfer` events are emitted when taking taxes from a transfer amount and sending to contract, and wallets for liquidity, marketing, development, and game rewards fees. Without a `Transfer` event, the movement of tokens can only be seen by checking balances of the accounts; it is otherwise an entirely blind movement of tokens and can't be tracked. `Transfer` events must be emitted even for zero value transfers to be compliant with the EIP-20 Token Standard.

Location	`_takeLiquidity`
	`_takeMarketing`
	`_takeDev`
	`_takeGameRewards`

Recommendation	Recommended to emit a `Transfer` event for tax transfers.
Resolution	Resolved by emitting a `Transfer` event for all fee transfers.

Issue #4	Incorrect Transfer Event Value
Severity	Medium Resolved
Description	If game reward taxes are greater than 0% and taken from a transfer, the `Transfer` event emits with incorrect transfer amount value. The actual tokens transferred are not affected, but the event emits with the total request amount, minus all fees except the game rewards fee. It is expected that the event would emit the total amount requested to transfer minus all fees, showing the actual value transferred to recipient.
Location	`_transferStandard`
Recommendation	Recommended to emit actual value transferred to recipient. This should be fixed by also resolving **Issue #12**. See that issue for more information.
Resolution	Resolved by fixing **Issue #12** to ensure all fees are removed from transfer value emitted in `Transfer` event.

Issue #5	Burn Function
Severity	Medium Resolved
Description	Contract contains a burn function which removes tokens from total supply without reducing total supply value or easily identifying burned amount. This will likely not cause any internal issues, but will lead to inaccurate results from calculations that utilize total supply. For example: market cap, percentage of total tokens held by a given address, etc.
Location	`burn`
Recommendation	Recommended to reduce total supply with burn and emit `Transfer` event to `address(0)`.
Resolution	Resolved by reducing total supply with burn and emitting `Transfer` to `address(0)`.

Issue #6	Liquidity Recipient
Severity	Medium Partially Resolved
Description	The `addLiquidity` function is called with contract owner specified as new LP recipient. As a result, if enough liquidity is generated over time, the contract owner will eventually hold a large portion of total liquidity.
Location	`addLiquidity`
Recommendation	Consider sending LP tokens to the token contract (or a custom contract) with added functions that can lock or otherwise restrict the tokens from being misused by a single entity. Alternatively, renouncing ownership of the contract will permanently lock the LP tokens in address(0). At the very least, LP tokens should be manually added to a lock as they are collected over time. This is less secure for investors and all investors should do their own research to ensure majority of liquidity is locked at all times.
Resolution	The team has committed to manually locking any large amount of generated LP over time.

3.4 Minor Findings (4)

Issue #7	Receive Function
Severity	Minor Resolved
Description	Contract contains a receive function which will accept BNB sent from any address.
Location	`receive`
Recommendation	Recommended to only allow router or owner address in receive function to avoid accidental BNB sent to contract.
Resolution	Resolved by limiting `receive` function to owner and router addresses.

Issue #8	Permanently Locked BNB
Severity	Minor Resolved
Description	Contract has no method defined to allow withdrawing BNB. Due to current finding in `receive` function (**Issue #7**), funds can be sent to contract and essentially burned since there is no method to retrieve. When `swapAndLiquify` is performed, there will also be some BNB left on contract which can't currently be withdrawn. This extra BNB is generated due to token price difference when swapping to BNB and then pairing tokens and BNB to add liquidity.
Location	`swapAndLiquify` `receive`
Recommendation	Recommended to add a `withdraw` function to withdraw accumulated BNB or have some other manually invoked functions to use the BNB such as adding liquidity without selling stored tokens or performing a buy back and burning purchased tokens.
Resolution	Resolved by fixing **Issue #7** and adding a `withdrawBNB` function to allow withdrawing any extra accumulated BNB from `swapAndLiquify` events.

Issue #9	Regaining Renounced Ownership
Severity	Minor Resolved
Description	Due to a logical issue with utilized `Ownable` contract, it is possible for contract owner to renounce or transfer the contract ownership and then reclaim ownership at a later date. This is performed by first locking ownership (temporarily renouncing to zero address), then calling `unlock` which checks if caller was previous owner, but does not reset previous owner value. This then allows the owner to renounce ownership or transfer ownership to another address, and then call the `unlock` function again at any point to regain ownership since they are still counted as previous owner until a `lock` event happens again.

Location	`contract Ownable`:
	`lock`
	`unlock`

Recommendation	Recommended to use OpenZeppelin Ownable implementation or similar. If ability to temporarily renounce contract is needed, the issue can be fixed by setting `_previousOwner = address(0)` in the `unlock` function. Alternatively, also consider using TimelockController from OpenZeppelin.
Resolution	Resolved by adding `_previousOwner = address(0);` in the `unlock` function.

Issue #10	Router Calls
Severity	Minor Resolved
Description	External calls to `swapExactTokensForETHSupportingFeeOnTransferTokens` and `addLiquidityETH` may fail and cause an investor's transfer to fail.
Location	`swapTokensForEth` `addLiquidity`
Recommendation	Recommended to wrap the external calls in a `try/catch` statement to ensure all transfers will succeed even if swap and liquify event fails.
Resolution	Resolved by wrapping external calls to `swapExactTokensForETHSupportingFeeOnTransferTokens` and `addLiquidityETH` in `try/catch` statements.

3.5 Informational Findings (10)

Issue #11	No State Change Events
Severity	Informational Resolved
Description	There are a number of state affecting functions in contract that do not emit an event.

Location	`excludeFromFee`
	`includeInFee`
	`excludeFromWhitelist`
	`includeInWhitelist`
	`setLiquidityFeePercent`
	`setMarketingFeePercent`
	`setDevFeePercent`
	`setGameRewardsFeePercent`
	`setMaxTxPercent`
	`setMaxTxAmount`

Recommendation	Recommended to always emit an event when state is updated by a function.
Resolution	Resolved by emitting events for all given events.

Issue #12	Game Rewards Fee Code
Severity	Informational Resolved
Description	In `_getTValues` function, there is code to take a game rewards fee that has been commented out and moved to parent calling function, probably to avoid a `Stack too deep` error. This is confusing scope wise for the two functions and could cause bugs if further token versions are developed from this contract code and developer is not aware of function responsibilities. In fact, this likely lead to **Issue #4**.
Location	`_getTValues` `_transferStandard`
Recommendation	Recommended to calculate all fees inside `_getTValues` function.
Resolution	Resolved by calculating all fees inside of `_getTValues` function.

Issue #13	Transfer Exceeded Balance
Severity	Informational Resolved
Description	Transfer function does not check sender balance before attempting to send. Send fails due to subtraction overflow instead of transfer amount exceeding balance.
Location	`_transfer`
Recommendation	Recommended to ensure sender has enough tokens before attempting to perform math on balances and return more specific `ERC20: transfer amount exceeds balance` error message.
Resolution	Resolved by explicitly checking sender balance and returning more specific error message.

Issue #14	Hardcoded Swap Threshold and Unused Event
Severity	Informational Resolved
Description	Variable `numTokensSellToAddToLiquidity` cannot be modified after contract deployment, and there is a related event `MinTokensBeforeSwapUpdated` which is not used. It may be beneficial to have ability to change swap amount if contract is selling too often. There is also the possibility, especially if massive burns happen in the future, that contract could be swapping too rarely and dumping the charts with a large sell.
Location	`numTokensSellToAddToLiquidity`
Recommendation	Consider adding a function to allow changing the value of variable and utilize existing event.
Resolution	Resolved by adding a function to allow changing the value of `numTokensSellToAddToLiquidity` and renaming and emitting the existing event.

Issue #15	Hardcoded Fee Wallets
Severity	Informational Acknowledged
Description	The contract contains hardcoded addresses for the wallets collecting fees for marketing, development, and game rewards taxes. While these addresses may never need to be changed, it may be desired to change them in the future, for example to send game rewards to a contract instead of the hardcoded wallet for better security. It is also possible the addresses could be compromised in the future and the only way to avoid losing collected fees with current design is to stop collecting them.
Location	`contract MetaArcade`
Recommendation	Consider adding functions to allow changing the fee collecting addresses.
Resolution	Acknowledged

Issue #16	Potential Swap on Regular Transfer
Severity	Informational Resolved
Description	The swap and liquify event sells contract owned tokens during a transfer event if the transfer is not a purchase from the token pool. By avoiding swapping tokens on purchases, no sells will be linked to purchase events, even if the sell is only the contract swapping tokens to create liquidity. That is as desired, but by only checking for a purchase from the pool, the contract does not check anything else, so a swap and liquify event may happen on other non-sell actions such as staking tokens or sending tokens to one of your other wallets. Some websites that read from the blockchain will accidentally show those non-sell transfers as a sell event, when it is only the contract selling its own tokens and not a sell event by the holder.
Location	`_transfer`
Recommendation	Recommended to limit swap and liquify event to happen on sells only.
Resolution	Resolved by checking transfer recipient is the token pair before initiating swap and liquify event.

Issue #17	Constant Values
Severity	Informational Resolved
Description	There are a number of variables that are never changed that could be declared as constants.
Location	`_tTotal`, `_name`, `_symbol`, `_decimals`, `numTokensSellToAddToLiquidity`
Recommendation	Recommended to declare the variables as constant or utilize `pure` getter functions where applicable
Resolution	Resolved by making `_name, _symbol, _decimals` into constants and convert the related `view` functions into `pure` functions. `_tTotal` and `numTokensSellToAddToLiquidity` are no longer constant due to resolutions for issues **#5** and **#14**

Issue #18	Public Functions
Severity	Informational Resolved
Description	The contract contains many `public` functions which can all be converted to `external` functions with very little change. `external` functions can not be called internally like `public` functions can leading to potentially cheaper execution.

Location	`name`
	`symbol`
	`decimals`
	`totalSupply`
	`balanceOf`
	`transfer`
	`allowance`
	`approve`
	`transferFrom`
	`increaseAllowance`
	`decreaseAllowance`
	`excludeFromFee`
	`includeInFee`
	`excludeFromWhitelist`
	`includeInWhitelist`
	`setSwapAndLiquifyEnabled`
	`isExcludedFromFee`
	`isWhitelisted`
	`burn`

Recommendation	Recommended to convert all `public` functions to `external` functions.
Resolution	Resolved by converting all listed `public` functions to `external` functions.

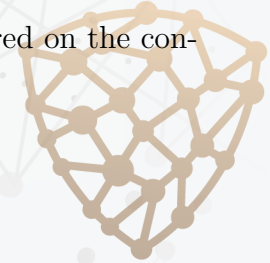
Issue #19	Order of Functions
Severity	Informational Resolved
Description	The contract orders functions in a semi-random order with some `external` functions coming after some `private` functions, etc. To help readers find information, it is recommended to group functions according to visibility in the following order: • `constructor` • `receive function` (if exists) • `fallback function` (if exists) • `external` • `public` • `internal` • `private`
Location	`contract MetaArcade`
Recommendation	Recommended to order all functions in `MetaArcade` contract per given ordering.
Resolution	Resolved by ordering all functions by visibility per given ordering.

Issue #20	Inadequate Documentation
Severity	Informational Acknowledged
Description	The majority of the contract contains no comments for documentation. It is good practice to fully annotate Solidity contracts using NatSpec Format for all public interfaces. Inline comments to document sections of the code as appropriate is also recommended.
Location	`contract MetaArcade`
Recommendation	Recommended to fully annotate using NatSpec format all public interfaces (everything in the ABI) and add inline comments for clarification if necessary.
Resolution	Acknowledged

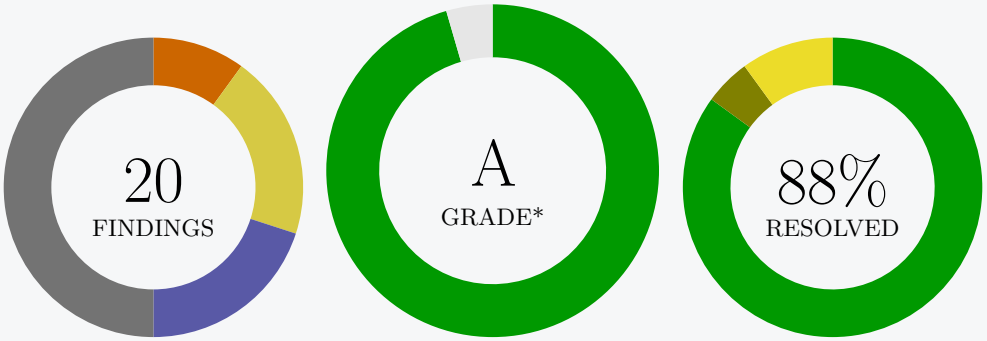
4 Owner Privileges

Owner privileges are not inherently an issue, and any known issues were presented and resolved in audit findings, but it is always good to be aware of the abilities a contract owner has over the contract state. The following functions are the available abilities in the final tested version of the audited contract that the owner of Meta Arcade token has while the contract is not renounced:

- `renounceOwnership` - The owner can renounce ownership, meaning none of the owner only functions can be successfully called anymore.
- `transferOwnership` - The owner can transfer ownership to a new address.
- `lock` - The owner can "lock" the contract, meaning they can't call any owner only functions until the specified unlock time.
- `unlock` - The owner can "unlock" the contract after lock time has elapsed, meaning they can once again call owner only functions.
- `excludeFromFee` - The owner can exclude any address from fees on transfers.
- `includeInFee` - The owner can include any address in fees on transfers.
- `excludeFromWhitelist` - The owner can exclude any address from whitelist, meaning they cannot make transfers greater than defined maximum transaction amount.
- `includeInWhitelist` - The owner can include any address in a whitelist which allows the address to make transfers of any size.
- `setLiquidityFeePercent` - The owner can set the liquidity fee to a maximum value of 10% for a total maximum tax of 30%.
- `setMarketingFeePercent` - The owner can set the marketing fee to a maximum value of 10% for a total maximum tax of 30%.
- `setDevFeePercent` - The owner can set the development fee to a maximum value of 10% for a total maximum tax of 30%.
- `setGameRewardsFeePercent` - The owner can set the game rewards fee to a maximum value of 10% for a total maximum tax of 30%.
- `setMaxTxPercent` - The owner can set the maximum transaction amount by providing desired percent greater than or equal to 1.
- `setMaxTxAmount` - The owner can set a maximum transaction amount greater than or equal to 0.1% of total supply.
- `setMaxWalletAmount` - The owner can set a maximum wallet of any size. Wallets larger than maximum will always be able to sell.
- `setSwapThreshold` - The owner can set the number of tokens the contract gathers before selling half and pairing for liquidity.
- `setSwapAndLiquifyEnabled` - The owner can enable or disable converting the tokens gathered from liquidity fee into liquidity by selling half of the tokens and pairing.
- `withdrawBNB` - The owner can withdraw any extra BNB gathered on the contract due to price fluctuations when converting to liquidity.



5 Summary



*Grade is not an endorsement or disapproval and is automatically calculated based on unresolved findings and their severity levels

The Meta Arcade token is an EIP-20 compliant liquidity generating token. The token can take fees from all transfers to generate liquidity or send tokens directly to marketing, development, and game reward wallets. As of writing, a 2% fee is taken for each of the 4 taxes for a total fee on transfers of 8%. Liquidity is generated by selling gathered tokens from the liquidity tax on a sell event when a swap threshold is reached.

Shield Network Auditing performed an audit of the Meta Arcade token contract through a process of manual line-by-line code review, testnet deployment and testing, and automated test review and development. A total of 20 findings were documented through the audit process.

There were no Critical findings in the audit and the majority of findings were Informational level. All Major, Medium, and Minor findings were resolved except for the medium finding of newly generated liquidity being sent to the owner wallet. This finding has been marked as Partially Resolved since the team has committed to locking any significant liquidity amount held by the owner wallet.

6 Revisions

This section documents any revisions made in the audit after initial public release. All revisions resulting from contract changes were made following the same audit methodologies used for the initial audit.

6.1 Revision 1

A new version of Meta Arcade token contract was deployed after initial audit release to add a maximum wallet amount feature. In this audit revision, the final contract address in the Project Overview section was changed and the new owner only function `setMaxWalletAmount` was documented in the Owner Privileges section.