

Audit Report

- ender,202



ZIZ.

For

/https://ampt.finance

Contents

Scope of Audit

Check Vulnerabilities

Techniques and Methods

01

01



Issue Categories	03
Number of security issues per severity.	03
Introduction	04
A. Contract - Amplify Child Token	05
High Severity Issues	05
1. Balance of senders is updated incorrectly	05
2. Deposit and Withdraw do not update vote balance	06
Medium Severity Issues	07
Low Severity Issues	80
3. Lack of Input Validation	80
Informational Issues	80
4. Typos	80
5. State Variable Default Visibility	80
6. State variables that could be declared immutable	09
7. Missing Events for Significant Transactions	09

Functional Tests

10

Automated Tests

Closing Summary



Scope of the Audit

The scope of this audit was to analyze and document the Amplify Child Token smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known

vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
 - Unchecked external call
 - Unchecked math
 - Unsafe type inference
 - Implicit visibility level







Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.



Gas Consumption In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.





Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart

	contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now.

Informational These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity



Open			
Acknowledged			
Closed	2	1	4





Introduction

During the period of **November 02, 2021 to November 09, 2021** -QuillAudits Team performed a security audit for **Amplify Child Token** smart contracts.

The code for the audit was taken from following the official link: https://github.com/amplifylabs/contracts/blob/main/protocol/contracts/

<u>Governance/AMPTChild.sol</u>

Ver. No.	Date	Commit hash
1	November 02	31526ea5e5c5a1c308d0065aab1f57ec8d765fc6
2	November 09	52262ff372119a73ae10970a7980d91510c00a43







Ssues Found

A. Contract – AMPTChild

High severity issues

1. Balance of senders is updated incorrectly



Description

The balance of the sender is updated incorrectly due to the operation at line 331 in the function _transferTokens().

L331: balances[src] -= safeSub(balances[src], amount, "AMPT::_transferTokens: transfer amount exceeds balance");

As we can see that the **balances[src] = balances[src]** safeSub(balances[src], amount), instead of balances[src] = safeSub(balances[src], amount).

Remediation

We recommend changing the operation above to

balances[src] = safeSub(balances[src], amount, "AMPT::_transferTokens: transfer amount exceeds balance");

Status: Fixed in version 02





2. Deposit and Withdraw do not update vote balance

Line	Code
94-105	<pre>function deposit(address user, bytes calldata depositData) external { require(msg.sender == childChainManagerProxy, "You're not allowed to deposit"); require(user != address(0), "AMPT::deposit: cannot transfer from the zero address");</pre>

```
      Zero address');

      uint256 amount = abi.decode(depositData, (uint256));

      // `amount` token getting minted here & equal amount got locked in

      RootChainManager

      totalSupply += amount;

      balances[user] += amount;

      emit Transfer(address(0), user, amount);

      }

      117-124

      function withdraw(uint256 amount) external {

      require(amount > 0, "Amount must be greater than 0");
```

balances[msg.sender] = safeSub(balances[msg.sender], amount, "ERC20: burn amount exceeds balance"); totalSupply = safeSub(totalSupply, amount, "AMPT::withdraw: update total suply failed");

emit Transfer(msg.sender, address(0), amount);





Description

The **deposit and withdrawal** functions work like mint and burn for ERC20 tokens. But in this token's implementation every transfer needs to call the **_moveDelegates** function to update the vote balance of the delegate.

Here the **withdraw** function does not update the vote balance of the delegates when the user burns its tokens. Similarly, the **deposit** function does not update the vote balance and delegates can lose their votes.

This can be exploited by executing the following loop several times : deposit \rightarrow delegate \rightarrow withdraw. This way any user can gain access to unlimited votes.

Remediation

We recommend,

Using the following in deposit function after updating the balance and total supply:

_moveDelegates(delegates[address(0)], delegates[user], amount);

Using the following in withdraw function after updating the balance and total supply: _moveDelegates(delegates[msg.sender], delegates[address(0)], amount);

Status: Fixed in version 02

Medium severity issues

No issues were found.





Low level severity issues

3. Lack of Input Validation

The newChildChainManagerProxy is not checked and compared with the current **childChainManagerProxy**.

We recommend adding a check if the **newChildChainManagerProxy == childChainManagerProxy**, which should revert the transaction, this helps to avoid the potential for erroneous values to result in unexpected behaviors or wasted gas.

Status: Fixed in version 02

Informational issues

4. Typos

Please consider fixing the following typos:

L121: suply should be supply

Status: Fixed in version 02

5. State Variable Default Visibility

L35: address deployer;

Description

The Visibility of the aforementioned variable is not defined. Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

Status: Fixed in version 02





6. State variables that could be declared immutable

Line	Code
35	address deployer;

Description

The above state variables should be declared immutable to save gas.

Remediation Add the immutable attributes to state variables that never change after deployment.

Status: Fixed in version 02

7. Missing Events for Significant Transactions

Description

The missing event makes it difficult to track off-chain decimal changes. An event should be emitted for significant transactions calling the functions :

- updateChildChainManager

Remediation We recommend emitting the appropriate events.

Status: Fixed in version 02





Functional Tests

Functio	Inputs	Output	Expected	Transaction	Result	Fix
ns			Output	Hash		Status
updateC hildChai nManag er	0x60b6D91c B698F41E1 eD928f9631 cEC6b8Ff8F 6cC	0x60b6D91c B698F41E1 eD928f9631 cEC6b8Ff8F 6cC	0x60b6D91c B698F41E1 eD928f9631 cEC6b8Ff8F 6cC	0x587c418c4eabfca c3ee5adbecc1a3e5 a778ef870429fdf080 5e087b1c9071c77	Passed	N/A
deposit	" $0x153b057$ d5d7262dC9 2099B59c97 5255ecE667 84F"," $0x0000000000000000000000000000000$	Tru e	Tru e	0xc96301c48ec3b8 766b736805fe23434 81913493c209f58ee 163457ef85c3c18a	Passed	N/A
transfer	"0x1dd6439 4E29c5988f 04A8E074D 0DBACd4D6 14729","69"	Tru e	Tru e	0xd8172016e04caa bff05608949c59a8f6 762dc4256432bd47 aad850f0f26af3d3	FAILE	Fixed
balance Of	0x153b057d 5d7262dC92 099B59c975 255ecE6678 4F	69	0	call0x153b057d5d7 262dC92099B59c97 5255ecE66784F0xA 4eceE6f0fD8f39991 D20B0B2c0b10c08a 072c6A0x70a08231 00000000000000000 00000000000000000 5d7262dc92099b59 c975255ece66784f	Passed	
balance Of	0x1dd64394 E29c5988f0	69	69		Passed	N/A





	DBACd4D61 4729					
approve	"0x1dd6439 4E29c5988f 04A8E074D 0DBACd4D6 14729","20"	True	True	0x2d043932204c84 7109062d9688c0d2 142f6dd3c5ab6d275 b617d9e087485a17 6	Passed	N/A
transfer From	"0x153b057 d5d7262dC9 2099B59c97 5255ecE667 84F","0x1dd 64394E29c5 988f04A8E0 74D0DBACd 4D614729"," 20"	True	True	0xf246125df92ed5e 2ecd351628bdd268f 551bdf131f931df0bc ab2658ffaa1ca6	Passed	N
approve	"0x1dd6439 4E29c5988f 04A8E074D 0DBACd4D6 14729","20"	True	True	0x695850a9b0456f7 27530380dbc34c85 0a5d3d99f51eb6ca4 bad5558cb67203a7	Passed	N/A
decreas eAllowa nce	"0x1dd6439 4E29c5988f 04A8E074D 0DBACd4D6 14729","15"	True	True	0x94b367d2c63894 77201ba924673612 3743852e30c9e365 92b784bc7003ea2a 94	Passed	N/A
allowan ce	0x153b057d 5d7262dC92 099B59c975 255ecE6678 4F, 0x1dd64394 E29c5988f0 4A8E074D0 DBACd4D61 4729 = 5	5	5	call0x153b057d5d7 262dC92099B59c97 5255ecE66784F0xA 4eceE6f0fD8f39991 D20B0B2c0b10c08a 072c6A0xdd62ed3e 00000000000000000000 00000000000000	Passed	





<section-header></section-header>	"0x1dd6439 4E29c5988f 04A8E074D 0DBACd4D6 14729","15"	True	True	0x003328bc7cb925 0b46047ec694c123 7ae9a5125eb3392f0 122db9208a0ed69b 8	Passed	N/A
allowan ce	0x153b057d 5d7262dC92 099B59c975 255ecE6678 4F, 0x1dd64394 E29c5988f0 4A8E074D0 DBACd4D61 4729	True	True		Passed	N/A
withdra	from: 0x1dd64394 E29c5988f0 4A8E074D0 DBACd4D61 4729 withdraw: 30	True	True	0x4dfb757778a6fa3 292cccafab5de1249 6dc36f07b1051aa52 93ce223929b973c	Passed	N/A





Automated Tests

Slither

INFO:Detectors:

AMPTChild._writeCheckpoint(address,uint256,uint256,uint256) (AMPTChild.sol#356-367) uses a dangerous strict equality: - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == currentBlockNumber (AMPTChild.sol#359) Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities INFO:Detectors: AMPTChild.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (AMPTChild.sol#253-263) uses timestamp for comparisons

- Dangerous comparisons:

- require(bool,string)(getBlockTimestamp() <= expiry,AMPT::delegateBySig: signature expired) (AMPTChild.sol#261)</p> Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

INFO:Detectors:

AMPTChild.getChainId() (AMPTChild.sol#384-388) uses assembly

– INLINE ASM (AMPTChild.sol#386)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

INFO:Detectors:

Pragma version0.8.4 (AMPTChild.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6 solc-0.8.4 is not recommended for deployment

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

INFO:Detectors:

- approve(address,uint256) should be declared external:
 - AMPTChild.approve(address, uint256) (AMPTChild.sol#142-145)
- increaseAllowance(address,uint256) should be declared external:
 - AMPTChild.increaseAllowance(address,uint256) (AMPTChild.sol#159-162)

decreaseAllowance(address,uint256) should be declared external:

– AMPTChild.decreaseAllowance(address, uint256) (AMPTChild.sol#177-183)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external INFO:Slither:AMPTChild.sol analyzed (1 contracts with 75 detectors), 8 result(s) found INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration

Mythril

==== Dependence on predictable environment variable ==== SWC ID: 120 Severity: Low Contract: 0x4FB7d2f0c2e673B063258b8550d260bAB020d2c2 Function name: getPriorVotes(address,uint256) PC address: 3301 Estimated Gas Usage: 1479 - 2045 A control flow decision is made based on a predictable variable. The block.number environment variable is used in to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block num ber and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use an

of those environment variables for random number generation or to make critical control flow decisions.

Initial State:

Account: [ATTACKER], balance: 0x0, nonce:0, storage:{} Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:





Account: [ATTACKER], balance: 0x1, nonce:0, storage:{} Account: [SOMEGUY], balance: 0x1000000000041, nonce:0, storage:{}

Transaction Sequence:

==== Potential denial-of-service if block gas limit is reached ====
SwC ID: 128
Severity: Low
Contract: 0x4FB7d2f0c2e673B063258b8550d260bAB020d2c2
Function name: gasprice_bit_ether(int128)
PC address: 11541
Estimated Gas Usage: 2061 - 2916
Potential denial-of-service if block gas limit is reached.
A storage modification is executed in a loop. Be aware that the transaction may fail to execute if the loop is unbounded and the necessary gas exceeds the block gas limit.

Initial State:

Account: [ATTACKER], balance: 0x1000000000001, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x40000040000, nonce:0, storage:{}

Transaction Sequence:

SOLHINT LINTER

<u>AMPTChild</u>	.sol		
2:1	error	Compiler version 0.8.4 does not satisfy the ^0.5.8 semver requirement	compiler-version
7:5	warning	Constant name must be in capitalized SNAKE_CASE	const-name-snakecas
10:5	warning	Constant name must be in capitalized SNAKE_CASE	const-name-snakecas
13:5	warning	Constant name must be in capitalized SNAKE_CASE	const-name-snakecas
35:5	warning	Explicitly mark visibility of state	state-visibility
71:5	warning	Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)	func-visibility
72:9	warning	Error message for require is too long	reason-string
80:9	warning	Error message for require is too long	reason-string
96:9	warning	Error message for require is too long	reason-string
179:9	warning	Error message for require is too long	reason-string
186:9	warning	Error message for require is too long	reason-string
187:9	warning	Error message for require is too long	reason-string
225:13	warning	Error message for require is too long	reason-string
258:9	warning	Error message for require is too long	reason-string
259:9	warning	Error message for require is too long	reason-string
261:9	warning	Error message for require is too long	reason-string
284:9	warning	Error message for require is too long	reason-string
328:9	warning	Error message for require is too long	reason-string
329:9	warning	Error message for require is too long	reason-string
374:16	warning	Avoid to make time-based decisions in your business logic	not-rely-on-time
386:9	warning	Avoid to use inline assembly. It is acceptable only in rare cases	no-inline-assembly

21 problems (1 error, 20 warnings)

Results

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.





Closing Summary

In this report, we have considered the security of the Amplify Child Token platform. We performed our audit according to the procedure described above.

The audit showed several high, low, and informational severity issues. In the end, all of the issues were fixed by the Auditee.





Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the **Amplify Child Token** platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the **Amplify Child Token** Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.





véníber 202 Π XÍ



ARTICIES

https://ampt.finance

Canada, India, Singapore, United Kingdom

audits,quillhash.com

Ø

 \mathbf{G}

QuillAudits

audits@quillbasb.com