

Goshen Network

Audit Report

VER 1.0

23rd August 2022

No. 2022082316231

Project Summary

1. Project Introduction

The smart contract of Goshen Network mainly realizes the batch submission of layer 2 transactions, the cross-layer communication between layer 1 and layer 2, the state machine, and the status challenge in the window period. Thus, the optimal rollup mechanism realizes the bulk upload of transactions to the main chain of layer1 after summary and packaging and ensures that anyone can fully deduce the status of layer2 by synchronizing the blocks of layer1. A large number of transactions can be packaged in a single rollup block, which can effectively improve the throughput of the main chain.

2. Audit Summary

Project Name	Goshen Network	Platform	N/A
Token	N/A	Token symbol	N/A
Start date	26th July 2022	Language	Solidity
End date	19th August 2022	Website	N/A
Github	https://github.com/ontology-layer-2/rollup-contracts/tree/e3634932ba55ffc256d72180306f091acdbd1fc4/contracts	Whitepaper	N/A

3. Audit Scope

ID	File	SHA-256 checksum
contracts/token	L2StandardERC20.sol	07e68f88722019c2c106d1cc4648704b894cc8431d4c68d44b39d10b7bb4ffda
contracts/test-helper	TestL2ERC20.sol	b101139238c6d2e6054255b682a43f58a3a518c18d38e43198f7811e8c26f4dc
contracts/test-helper	depends.sol	4cf43192eb98acfe122735dbe4d1c639b2c1a10840c81e9763e33bc58acff2c1
contracts/test-helper	TestBase.sol	f8431a13c160e019ae98191b44eb5d0b70254daf8129569c5942c85c0b6ab8e8
contracts/test-helper	TestERC20.sol	3238ec1c309f87afc2c5c8c25f29379c364b332b990f09c25c4fbb44ba47d1f1
contracts/state-machine	MemoryLayout.sol	b43d3ff99f0f104132551704934ce33c72797395809026d5bc9aa3756738aaf3
contracts/state-machine/riscv32	Instruction.t.sol	f3e4154044666f21816fab68ba17bfb9d0f667a69b49f0f1ad2ea037c0366f44
contracts/state-machine/riscv32	Instruction.sol	ec26eae1754f3ab3ee18221994ef49c84e21470be748bce5fe73d2e2f11ec661
contracts/state-machine/riscv32	Register.sol	ee01e3387317f17cf6951b27eea93c3533c053c535fe7e669b015161986c4283
contracts/state-machine/riscv32	Interpreter.t.sol	7bda315fa3d2bea5a0f18ed74dfac0d614263685f3276b179bda728653379d88
contracts/state-machine/riscv32	Interpreter.sol	ec9dd279507c8af11731b4f1760b9124a82cd84bbf76cc3c95a5eba27f80182a
contracts/state-machine/riscv32	Syscall.sol	9d3cb83830ccb2f5ec393cca29d967a638d9b96b87748b0f18dbeb371bc1febe
contracts/state-machine	Memory.t.sol	d1265a1317eb34f92761981e6c3ff3aa49936833d6a37ea6e1762424415263de
contracts/state-machine	StateTransition.sol	3b313ed08b0e5921f5a701c4ff924f849e870a35232a3e12e05291474ef289c7
contracts/state-machine	MachineState.sol	f1cca9fc6efb28ff78b060d4e0bcc84f208a8b2e42a1abe80b5d24c37a98a947
contracts/state-machine	Memory.sol	56f9f95eac68ddb0a8f13bf20f641c7ff07dceee883a424cd3e3e7709de5936f
contracts/bridge	L1StandardBridge.t.sol	98e41a5fcb84ac0256eab1354724c6b500e43d3bf b6f9786463be98c84248b4e
contracts/bridge	L1StandardBridge.sol	41899439179b6b24f04f4a06e5ddeb0420c38a9f7e3755bfb902eaaaa9fcf854
contracts/bridge	L2StandardBridge.t.sol	b8f1c37721222c6aba38a9ab96191f8af74de916d72d49f1a72224bdd099c2ff
contracts/bridge	L2StandardBridge.sol	6d262e6f8aad79e4f8451d39bf3501d03ade150e262953cc36938b4d20f232cd

ID	File	SHA-256 checksum
contracts/dao	Whitelist.sol	6c8b76c1d211d1725906f7466475f99c9ae243d91f8884da78d85f33ebbd9a27
contracts/libraries	MerkleTrie.t.sol	bd92acf778b265007c4cd8dfd47bf71f74d8de411c967bd1f12db7c7bbcdbfd9
contracts/libraries	BytesSlice.sol	4affb85dd19aaaf2e0816b6af020f5abc63c91cc4bf3d0fe3b3eca79089df2a4
contracts/libraries	UnsafeSign.sol	eabb4eb172d9ccea46f95f7ca95f0dc6d40baff6bc6af7cc9256dc40637f7091
contracts/libraries	RLPReader.sol	0bf8185099c06de88f2eb27946f627543b3b27dfd23435ca321670467557adb2
contracts/libraries	RLPWriter.t.sol	e69d9ce138245c3f3e7cb1251d4a7369ad8da4dd26acfa6dee517ee9fe1b2abf
contracts/libraries	RLPWriter.sol	608017eb9edf016dda0c3158d802351e7f2d8a7edaab7251c0e9c4371e4a746f
contracts/libraries	UnsafeSign.t.sol	41888a33f949b9aa853d6cd54cb4afcd1a5aa5bc5d39bf8af58774bf527f21ce
contracts/libraries	Constants.t.sol	edf24f96356fd4584e641f6e0e577a7f9e4be162df422d6e0e3f525c550f95e3
contracts/libraries	BytesEndian.sol	c65dc4496c31d29d1040a8fa2f1cded4939839cb0e2d83c85fbbc0e2ebbbf757
contracts/libraries	RLPCodec.t.sol	bf85698033b4e2499f8cd1b4d6784204e80766044546ca1bd58a77455f0bb938
contracts/libraries	BytesSlice.t.sol	3c7bbec3ecbc4dc18bafd3802cd44dcb0f69075d35bd57dfbdf82b6737ec5932
contracts/libraries	Constants.sol	01722c5f7eb2cc29082c6f1a924e9bbcd8852c3090e652aad7c30126a0b4d0db
contracts/libraries	Types.t.sol	146b374e48b3d5b273846426224f1aba9f0bd4a6f81a51465984a04869b92fba
contracts/libraries	MerkleMountainRange.sol	ea39edd7da279465f1bd85f8835aa77c3136607d333d322eac1f6af7203cc5be
contracts/libraries	console.sol	9253a62c35b652b253569aca3d8ccda142283ac392f83c89cc33bba450de9e38
contracts/libraries	MerkleTrie.sol	1517c4998015eb346fc524e90036be7a132765d4c6ccaf013fc3588dc31f053e
contracts/libraries	Types.sol	029fb9327db52dd5d0533b48951b2ddbcd7d6117d6b837a9d3b40bb904d0aca2
contracts/libraries	MerkleMountainRange.t.sol	84fcea91b96aae89be6e576569ad73b062c95451d4db1deea04341bf215ee833
contracts/builtins	L2FeeCollector.sol	a8062e67f84a62a667853d9ede8caffed2be1c0041f64dd4b75e5dfa3e7e1d6e
contracts/builtins	L2FeeCollector.t.sol	8344916317740f00498d0b6054dd30df8cc9c6b2e66d309d436efed246ec3bd1
contracts/staking	StakingManager.t.sol	6adac41ba145039b717ad6ea74051069c2b379d5901e180dc56fcf0dabb9c9e5
contracts/staking	StakingManager.sol	73ec5d04c5f305e8039efe889c151f9dfbae000cb633a59ccd64a43b37c61881

ID	File	SHA-256 checksum
contracts/rollup	RollupStateChain.t.sol	c70d0fbefbf7723996d039f89e5080b3e36daddbce5f5ce47abd0972fd4ac412d
contracts/rollup	RollupInputChain.sol	fa581ba6a780959ca22548c279eb5a68986b13ea89565f2242669f339163d50f
contracts/rollup	ChainStorageContainer.sol	00d89b898075f21f04777cb8f5fba275f141e4ddf3fd06f7df9336623aa8c03e
contracts/rollup	RollupInputChain.t.sol	0ed146d20622a3ebe0d421b1fb5613edda94c8eca0f86a716a02e81ca5d0f344
contracts/rollup	RollupStateChain.sol	174c3c470b5ef33dc73df8a841fc54b2408e3931f59e754121b20be349bb5af7
contracts/rollup	ChainStorageContainer.t.sol	435a1a30c4243631805f3b76b78a10afd6e1c776abc1c226bf1490c781e64d13
contracts/cross-layer	L1CrossLayerWitness.sol	9c2c2f8a260422b335b5e8cf41440c2d19db6721a3fd429da32ec0a84ad90b95
contracts/cross-layer	CrossLayerCodec.sol	90e4efbfb3fc464c6f9fcdee0d9c20d028d1281421fc036f6c66db211b1c4a6b
contracts/cross-layer	CrossLayerContext.sol	6b99f23b2e21ff28735c950e80464b393adead415af16bf91ab64351a2fa3fd2
contracts/cross-layer	L1CrossLayerWitness.t.sol	357e28bc671d35fb25740157aea20eb03aa5d0a2e9737de26a0bd6a5120cb8c9
contracts/cross-layer	CrossLayerWitness.t.sol	10f87db907c3e6e613042d5bc7df6becbfee87417fccecf4b046a9c5beb3bb5a
contracts/cross-layer	L2CrossLayerWitness.sol	eb1ae87a9408fb8f62cc38979b816fe713253312ef435bbd7a86146d765b3eeb
contracts/challenge	DisputeTree.t.sol	a2d117734c12d0fc444706b0860493b497cf43e566e50a5c2d124842c5af05cc
contracts/challenge	ChallengeFactory.sol	1f293f10782f9c5826f7065ebf280eef7c031251488b63725e4ab4b25166d830
contracts/challenge	Challenge.sol	02bd596d0d4a72198da306165abeac0194a24d0a68f470738fd971fa125bbc4c
contracts/challenge	ChallengeFactory.t.sol	3631db453463545f711831783b50b546d4ee9e424466e333ffc09f8b140ecf3
contracts/challenge	DisputeTree.sol	d4049a4a5d26d1c11554a799845a52fec8328b32f590b6bc7c717b97c8a6e38
contracts/resolver	AddressManager.sol	6f5f3b6823fdda4710e4e17ffd14b06948393ac49bf30ea68ae5344184042e20
contracts/resolver	AddressName.sol	044250e80b108dbe30bbe1534b2de822bbe9899222db968461d64bf771a3fc63
contracts/interfaces	IStakingManager.sol	776f50eadfd8f064d1e61697d7a6c347bf7a084b57f708310ecbc040f2daa25e
contracts/interfaces	IWhitelist.sol	824b7551d4b63e2698d909c820202a25fdd206c4d5f6a1c4b70e015d3939368f
contracts/interfaces	IRollupStateChain.sol	ddbaf3fac20fe8ee90a33ebbf535b51ae3f64f06db242aefd81e05ace6bdcd58
contracts/interfaces	IChainStorageContainer.sol	bee37816350c80a69b181ac105ce593990ee9e921b2bc95c42c65d8f5f2447b8

ID	File	SHA-256 checksum
contracts/interfaces	IL1StandardBridge.sol	8bd0bdb642863edaa4b64f8beadd39b4398b903213eeef0083e63a61ca743fa9
contracts/interfaces	IRollupInputChain.sol	1521df9a461369c50d72f772382884bc296d8ac8fb39afd3c21cf56df593d815
contracts/interfaces	IL2CrossLayerWitness.sol	cf2d40b00d9e2a6ff581d6499eb525d3589c39c30a111552206d9de0590e290f
contracts/interfaces	IAddressManager.sol	4641f578f115086690b798bef5abc43e24d7f1479ffe773311cc734333c5f3c8
contracts/interfaces	IAddressResolver.sol	28ab5b2f851b757b0145feb3842b41eedf2deb31c8d64277488f34361466c97c
contracts/interfaces	IL2ERC20Bridge.sol	9f26c185bfe020370dcb45d3e1dc344f6265af1185f0ee1c3258b4b402fdbcb32
contracts/interfaces	IChallengeFactory.sol	a7f88515fb211d42525ed7862eca97c9971778ac6a7c8a5890c41e48e0388a68
contracts/interfaces	IL1CrossLayerWitness.sol	6ff29d2b7dc0a56a3a56a89ea0ba33e7957536d61163bf302916dde4ea376325
contracts/interfaces	ICrossLayerWitness.sol	f16d6cbcc5b39e73edc170a2783987211882252530b1161feaa10f4f0dc36a92
contracts/interfaces	ForgeVM.sol	79322d85a0da847db26b9dcd83a0b077a3be979faad5a3114e9494c672a97bb2
contracts/interfaces	IL1ERC20Bridge.sol	fd8c586b3fb64d03cb8fecea5b801407789c709bd6669f106ef4657690b045cd
contracts/interfaces	IInterpretor.sol	32e6bcb4961b1d5b0b3d854e7c524f82015705ec830fa09dcfa42cc453324d96
contracts/interfaces	IStateTransition.sol	2f30bf12dea8ad11657f16b59b20d178f4e1c9c25c7358bec22acee6056816f4
contracts/interfaces	IMachineState.sol	87f48aa1f74afa38c0781a5a16a2e8fcd60db692ed548016295d8fd5e106afbb
contracts/interfaces	IChallenge.sol	347de9db7a59dbb09d61102c9d122e66aa04e35813bfd667b62f62aa74fa7fe0
contracts/interfaces	IL2StandardERC20.sol	2c74f738de908a74832991bd8163bdcd862286223cc578b6faecabbc4e4529af

4. Code Structure

contracts

├── bridge

| ├── L1StandardBridge.sol

| ├── L1StandardBridge.t.sol

| ├── L2StandardBridge.sol

| └── L2StandardBridge.t.sol

├── builtins

| ├── L2FeeCollector.sol

| └── L2FeeCollector.t.sol

├── challenge

| ├── Challenge.sol

| ├── ChallengeFactory.sol

| ├── ChallengeFactory.t.sol

| ├── DisputeTree.sol

| └── DisputeTree.t.sol

├── cross-layer

| ├── CrossLayerCodec.sol

| ├── CrossLayerContext.sol

| ├── CrossLayerWitness.t.sol

| ├── L1CrossLayerWitness.sol

| ├── L1CrossLayerWitness.t.sol

| └── L2CrossLayerWitness.sol

└── dao

| └── Whitelist.sol

├── interfaces

| ├── ForgeVM.sol

| ├── IAddressManager.sol

| ├── IAddressResolver.sol

| ├── IChainStorageContainer.sol

| ├── IChallenge.sol

| ├── IChallengeFactory.sol

| ├── ICrossLayerWitness.sol

| ├── IInterpreter.sol

| ├── IL1CrossLayerWitness.sol

| ├── IL1ERC20Bridge.sol

| ├── IL1StandardBridge.sol

| ├── IL2CrossLayerWitness.sol

| └── IL2ERC20Bridge.sol

#Standard bridge implementation

#Fraud challenge

Inter-layer communication

#Interface document

contracts

- | |—— IL2StandardERC20.sol
- | |—— IMachineState.sol
- | |—— IRollupInputChain.sol
- | |—— IRollupStateChain.sol
- | |—— IStakingManager.sol
- | |—— IStateTransition.sol
- | |—— IWhitelist.sol

—— libraries

- | |—— BytesEndian.sol
- | |—— BytesSlice.sol
- | |—— BytesSlice.t.sol
- | |—— Constants.sol
- | |—— Constants.t.sol
- | |—— MerkleMountainRange.sol
- | |—— MerkleMountainRange.t.sol
- | |—— MerkleTrie.sol
- | |—— MerkleTrie.t.sol
- | |—— RLPCodec.t.sol
- | |—— RLPReader.sol
- | |—— RLPWriter.sol
- | |—— RLPWriter.t.sol
- | |—— Types.sol
- | |—— Types.t.sol
- | |—— UnsafeSign.sol
- | |—— UnsafeSign.t.sol
- | |—— console.sol

—— resolver

- | |—— AddressManager.sol
- | |—— AddressName.sol

—— rollup

- | |—— ChainStorageContainer.sol
- | |—— ChainStorageContainer.t.sol
- | |—— RollupInputChain.sol
- | |—— RollupInputChain.t.sol
- | |—— RollupStateChain.sol
- | |—— RollupStateChain.t.sol

#Library files

#Packaging of rollup chain and state

contracts

- └── staking
 - └── StakingManager.sol
 - └── StakingManager.t.sol
- └── state-machine
 - └── MachineState.sol
 - └── Memory.sol
 - └── Memory.t.sol
 - └── MemoryLayout.sol
 - └── StateTransition.sol
 - └── riscv32
 - └── Instruction.sol
 - └── Instruction.t.sol
 - └── Interpretor.sol
 - └── Interpretor.t.sol
 - └── Register.sol
 - └── Syscall.sol
- └── test-helper
 - └── TestBase.sol
 - └── TestERC20.sol
 - └── TestL2ERC20.sol
 - └── depends.sol
- └── token
 - └── L2StandardERC20.sol

#State machine implementation

Audit Report Summary

1. Audit Methods

The audit was conducted to gain a clear understanding of how the project was implemented and how it works. The audit team conducted in-depth research, analysis, and testing of the project code and collected detailed data. In this report, the audit team will list in detail each issue identified, where it is located, the root cause of the issue, and a description of the issue, and will recommend changes to the issue accordingly.

Audit methods	Static analysis, Manual Review
---------------	--------------------------------

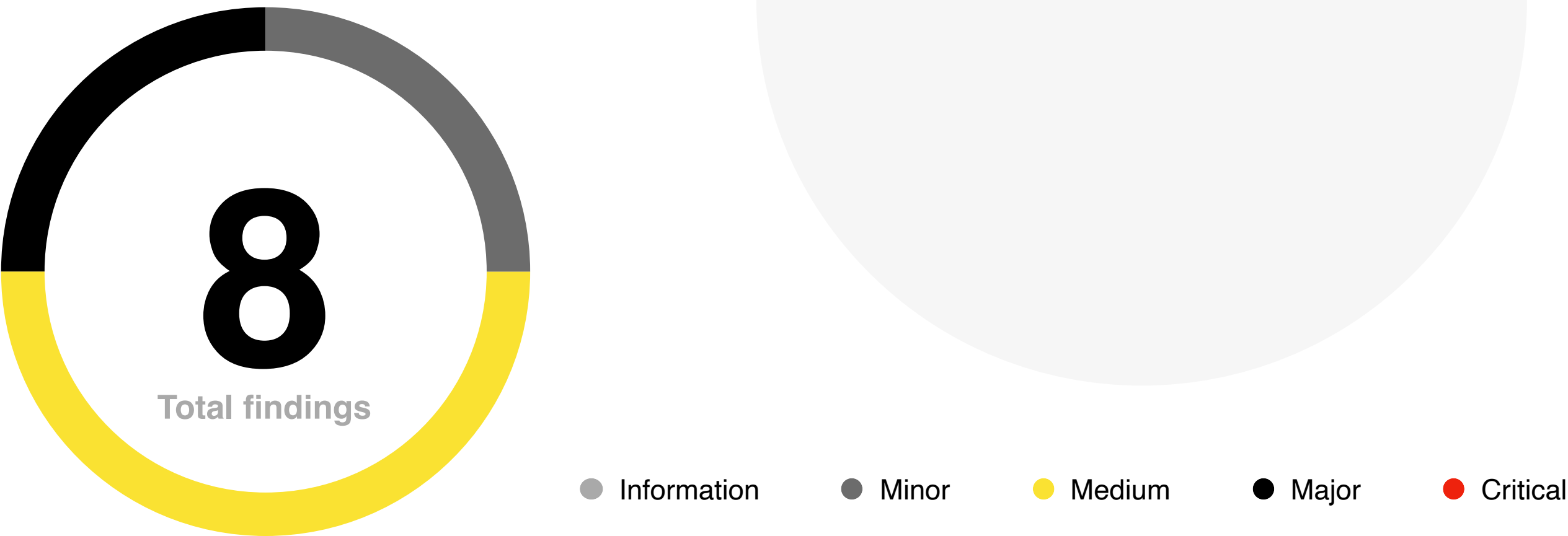
2. Audit Process

Steps	Operation	Description
1	Background	Read project descriptions, white papers, contract source code, and other relevant information the project team provides to ensure a proper understanding of project functions.
2	Automated testing	Scanning source code mainly with automated tools to find common potential vulnerabilities.
3	Manual reveiw	Engineers read the code line by line to find potential vulnerabilities.
4	Logical proofread	The engineer will compare the understanding of the code with the information provided by the project and check whether the code implementation is in line with the project white paper information.
5	Test case	Including test case design, test scope analysis, symbolic execution, etc.
6	Optimization items	Review of projects in terms of maintainability, safety, and operability based on application scenarios, deployment methods, and latest research results.

3. Risk Levels

Risk level	Issue description
Critical	Fatal risks and hazards that need to fixed immediately.
Major	Some high risks and hazards that will lead to related problems that must be solved
Medium	Some moderate risks and pitfalls may lead to potential risks that will eventually need to be addressed
Minor	There are low risks and hazards, mainly details of various types of mishandling or warning messages, which can be set aside for the time being
Information	Some parts can be optimized, such problems can be shelved, but it is recommended that the final solution

4. Audit Results



ID	Audit project	Risk level	Status
1	Reentrancy	None	
2	Injection	None	
3	Authentication bypass	None	
4	MEV Possibility	None	
5	Revert	None	
6	Race condition	None	
7	Insufficient Gas Griefing	None	
8	The major impact of flash loans	None	
9	Unreasonable economic model	None	
10	Predictable random numbers	None	
11	Voting rights management confusion	None	

ID	Audit project	Risk level	Status
12	Privacy leak	None	
13	Improper use of time on chain	None	
14	Improper codes in fallback function	None	
15	Improper identification	Major	Acknowledged
16	Inappropriate opcode	None	
17	Inappropriate assembly	None	
18	Constructor irregularities	None	
19	Return value irregularity	None	
20	Event irregularity	None	
21	Keywords irregularity	None	
22	Not following ERC standards	None	
23	Irregularity of condition judgment	Major	Acknowledged
24	Risk of liquidity drain	None	
25	Centralization Risk	None	
26	Logic change risk	None	
27	Integer overflow	None	
28	Improper function visibility	None	
29	Improper initialization of variables	None	
30	Improper contract calls	None	
31	Variable irregularities	None	
32	Replay	None	
33	Write to Arbitrary Storage Location	None	
34	Honeypot logic	None	
35	Hash collision	None	
36	Improper logic in receiving awards	None	
37	Use the not recommended method	None	
38	Basic coding principles were not followed	Minor	Acknowledged
39	Outdated external dependencies	None	
40	Business logic problems	Medium	Acknowledged

*In the above table, if the status column is “**Acknowledged**”, the audit team has informed the project owner of the vulnerability. Still, the project owner has not made any changes to the vulnerability or has not announced to the audit team the progress of the changes to the vulnerability. If the status column is “**Resolved**”, the project owner has changed the exposure, and the audit team has confirmed the changes.

5. Risk and Modification Program

The following section provides detailed information about the risk items learned after the audit, including the type of risk, risk level, location of the issue, description of the problem, recommendations for changes, and feedback from the project owner.

1. Business logic problems

Location	Contract file	Risk Status	Risk level
Line77-Line80, Line88-Line110	StakingManager.sol	⚠ Acknowledged	Medium

① Description

After the proposer is verified to be evil, the information in `getStakingInfo[proposer]` has not changed except that stakingstate is set to(line88-line110). The proposer can set the stakingstate of his / her account to holding again through deposit, and then do evil again. At this time, when the Challenger launch an activity, it will not be able to pass the required check (line 77-80)

```
proposerStake. earliestChallengeHeight == 0 || _ chainHeight < proposerStake.
earliestChallengeHeight
```

and cannot challenge.

② Recommendation

It is recommended to reset all parameters in proposerStack.

③ Code

Solidity/**

```
function claim(address _proposer, Types.StateInfo memory _stateInfo) external override
{
    StakingInfo storage proposerStake = getStakingInfo[_proposer];
    //only challenge.
    require(challengeFactory.isChallengeContract(msg.sender), "only challenge contract
permitted");
    require(proposerStake.state == StakingState.SLASHING, "not in slashing");
    require(rollupStateChain.verifyStateInfo(_stateInfo), "incorrect state info");
    _assertStateIsConfirmed(proposerStake.earliestChallengeHeight, _stateInfo);
    require(_stateInfo.blockHash != proposerStake.earliestChallengeBlockHash, "unused
challenge");
    token.transfer(msg.sender, price);

    //@OKLink Audit Description: Only state was reset, and no other information was
reset.
    //@OKLink Audit Solution: Reset all parameters in proposerStack.


    proposerStake.state = StakingState.UNSTAKED;
    emit DepositClaimed(_proposer, msg.sender, price);
}

function claimToGovernance(address _proposer, Types.StateInfo memory _stateInfo)
external override {
    StakingInfo storage proposerStake = getStakingInfo[_proposer];
    require(proposerStake.state == StakingState.SLASHING, "not in slashing");
    require(rollupStateChain.verifyStateInfo(_stateInfo), "incorrect state info");
    _assertStateIsConfirmed(proposerStake.earliestChallengeHeight, _stateInfo);
    require(_stateInfo.blockHash == proposerStake.earliestChallengeBlockHash, "useful
challenge");
    token.transfer(DAOAddress, price);

    //@OKLink Audit Description: Only state was reset, and no other information was
reset.
    //@OKLink Audit Solution: Reset all parameters in proposerStack.

    proposerStake.state = StakingState.UNSTAKED;
    emit DepositClaimed(_proposer, DAOAddress, price);
}
```


2. Irregularity of condition judgment

Location	Contract file	Risk Status	Risk level
Line40	ChallengeFactory.sol	 Acknowledged	Major

① Description

In the newChallenge function, the judgment condition of the require statement is incorrect, resulting in the failure to create a new challenge and the subsequent process.

② Recommendation

It is suggested to change code `<require(challengedStates[_hash] != address(0), "already challenged") >` to `<require(challengedStates[_hash] == address(0), "already challenged")>`

③ Code

Solidity/**

```
function newChallenge(
    //when create, creator should deposit at this contract.
    Types.StateInfo memory _challengedStateInfo,
    Types.StateInfo memory _parentStateInfo
) public {
    require(resolver.dao().challengerWhitelist(msg.sender), "only challenger");
    bytes32 _hash = _challengedStateInfo.hash();

    //@OKLink Audit Description: When creating a new challenge, the parameter of
    challengedStates[_hash] is 0.
    //@OKLink Audit Solution: require(challengedStates[_hash] == address(0),
    "already challenged");

    require(challengedStates[_hash] != address(0), "already challenged");
    require(resolver.rollupStateChain().verifyStateInfo(_challengedStateInfo),
    "wrong stateInfo");
    require(!resolver.rollupStateChain().isStateConfirmed(_challengedStateInfo),
    "state confirmed");
    require(resolver.rollupStateChain().verifyStateInfo(_parentStateInfo), "wrong
    stateInfo");
    require(_parentStateInfo.index + 1 == _challengedStateInfo.index, "wrong parent
    stateInfo");
    bytes32 _inputHash =
    resolver.rollupInputChain().getInputHash(_challengedStateInfo.index);
    bytes32 _systemStartState = resolver.stateTransition().generateStartState(
        _inputHash,
        _challengedStateInfo.index,
        _parentStateInfo.blockHash
    );
    ...
}
```


3. Improper identification

Location	Contract file	Risk Status	Risk level
Line55	ChallengeFactory.sol	⚠ Acknowledged	Major

① Description

When a Challenger conducts a fraud challenge, it creates a new challenge through the new challenge interface of the ChallengeFactory contract and calls the create function of the Challenge contract. At the same time, the challenger needs to deposit some tokens and recharge from the creator account to the corresponding Challenger's BeaconProxy contract address through transferFrom. Because the BeaconProxy contract was generated during the call of the function newChallenge and the transferFrom call occurred immediately, the BeaconProxy contract was not approved in the process, resulting in the transferFrom failure.

② Recommendation

It is recommended to call approve for authorization and allow the BeaconProxy contract to have the function of transferring user tokens

③ Code

Solidity/**

```
function newChallenge(
    //when create, creator should deposit at this contract.
    Types.StateInfo memory _challengedStateInfo,
    Types.StateInfo memory _parentStateInfo
) public {
    ...
    bytes memory _data;
    address newChallenge = address(new BeaconProxy(challengeBeacon, _data));
    challengedStates[_hash] = newChallenge;
}

//@OKLink Audit Description: The BeaconProxy contract is created for the
current function, and transferFrom is called without approval
//@OKLink Audit Solution: Call approve for authorization and allow the
BeaconProxy contract to have the function of transferring user tokens

IChallenge(newChallenge).create(
    _systemStartState,
    msg.sender,
    blockLimitPerRound,
    _challengedStateInfo,
    challengerDeposit
);
...
}
```

4. Business logic problems

Location	Contract file	Risk Status	Risk level
Line260-Line275	Challenge.sol	⚠ Acknowledged	Medium

① Description

After the Challenger succeeds in the fraud challenge, if there are multiple challengers participating, the rewards shall be distributed according to the tree root from top to bottom. The code implementation of the award allocation algorithm is inconsistent with the description in the comments, and it is necessary to consider whether the sum of the award allocation algorithm can allocate all the award funds. If the award funds cannot be allocated, the operation function for the remaining award funds needs to be added.

② Recommendation

It is suggested to modify the implementation of the reward allocation algorithm.

③ Code


Solidity/**

```
function _divideTheCake(
    uint256 _lowestNodeKey,
    uint64 _depth,
    address _challenger,
    IERC20 token
) internal {
    require(lastSelectedNodeKey[_challenger] != 0, "you can't eat cake");
    require(rewardAmount > 0, "no cake");
    uint256 _canWithdraw = minChallengerDeposit;
    uint64 _amount = _depth;
    //pay back deposit
    //  $vi = (i+k) / [n*(n+1)/2 + nk]$  ,  $k = 10$ ,  $n = 50$ ,  $v0 = 10/(25*51+ 500) = 1/355$ ,  $vn/v0 = 6$ 
    uint256 _scale; // @audit need to check wether the summer equals total
    uint256 _k = 10;
    uint256 _pieces = (((1 + _amount) * _amount) / 2) + (_amount * _k);
    uint256 _correctNodeKey = _lowestNodeKey;
    while (_correctNodeKey != 0) {
        DisputeTree.DisputeNode storage node = disputeTree[_correctNodeKey];
        //first pay back, and record the amount of gainer.
        if (_challenger == node.challenger) {
            _scale += (_amount + _k) / _pieces;
        }
        _amount--;
        if (node.parent == _correctNodeKey) {
            //reach the root
            break;
        }
        _correctNodeKey = node.parent;
    }

    // @OKLink Audit Description: The algorithm of reward allocation is inconsistent
    // with the annotation
    // @OKLink Audit Solution: Implementation of modified reward allocation
    // algorithm

    _canWithdraw += _scale * rewardAmount;
    lastSelectedNodeKey[_challenger] = 0;
    require(token.transfer(_challenger, _canWithdraw), "transfer failed");
}
```

5. Business logic problems

Location	Contract file	Risk Status	Risk level
Line69-Line90	Challenge.sol	 Acknowledged	Medium

① Description

Any user can call the create function of the Channel contract multiple times. The Challenge contract does not limit that the create function can only be called once, which enables the state of the Challenge contract to be reset by any external call. On the premise that the approved amount is sufficient, the token of the original creator can be transferred into the Challenge contract.

② Recommendation

It is suggested to add a stage judgment: `require(stage == ChallengeStage.Uninitialized, "only uninitialized stage");`


③ Code

Solidity/**

```
function create(
    bytes32 _systemStartState,
    address _creator,
    uint256 _proposerTimeLimit,
    Types.StateInfo memory _stateInfo,
    uint256 _minChallengerDeposit
) external override {
    //@OKLink Audit Description: The status of the current contract is not judged
    //@OKLink Audit Solution: require(stage == ChallengeStage.Uninitialized, "only
uninitialized stage");

    factory = IChallengeFactory(msg.sender);
    IERC20 depositToken = factory.stakingManager().token();
    systemInfo.systemStartState = _systemStartState;
    creator = _creator;
    proposerTimeLimit = _proposerTimeLimit;
    expireAfterBlock = block.number + proposerTimeLimit;
    systemInfo.stateInfo = _stateInfo;
    minChallengerDeposit = _minChallengerDeposit;
    //maybe do not need to deposit because of the cost create contract?
    require(depositToken.transferFrom(_creator, address(this),
minChallengerDeposit), "transfer failed");
    //started
    stage = ChallengeStage.Started;
    //emit by challengeFactory
    //emit ChallengeStarted(_blockN, _proposer, _systemStartState, _systemEndState,
expireAfterBlock);
}
```

6. Business logic problems

Location	Contract file	Risk Status	Risk level
Line95	RLPWriter.sol	 Acknowledged	Medium

① Description

In the `writelength` function, `encoded[0] = bytes1 (uint8 (lenLen + _offset + 55))` computationally converts the result to overflow, resulting the same `encoded[0]` output when executing functions in `writeUint` , `writeString` and `writeAddress`

Example 1: `lenlen=74, _offset=128` and `lenlen=330, _offset=128` , the results are all `encoded[0]=0x01`

Example 2: `lenlen=10, _offset=192` and `lenlen=266, _offset=192` , all results are `encoded[0]=0x01`

In optimization, the calculation method is: `bytes1 (uint8 (_len) + uint8 (_offset) +55)` ; For parameters that do not meet the requirements and exceed the bytes1 range limit, an error will be returned.

② Recommendation

It is suggested to modify as follows:`bytes1 (uint8 (_len) + uint8 (_offset) +55)` .

③ Code

Solidity/**

```
function _writeLength(uint256 _len, uint256 _offset) private pure returns (bytes
memory) {
    bytes memory encoded;


    if (_len < 56) {
        encoded = new bytes(1);
        encoded[0] = bytes1(uint8(_len + _offset));
    } else {
        uint256 lenLen;
        uint256 i = 1;
        while (_len / i != 0) {
            lenLen++;
            i *= 256;
        }
        encoded = new bytes(lenLen + 1);

        //@OKLink Audit Description: Use uint8 for forced conversion
        //@OKLink Audit Solution: Modify as bytes1(uint8(_len) + uint8(_offset)+55)

        encoded[0] = bytes1(uint8(lenLen + _offset + 55));
        for (i = 1; i <= lenLen; i++) {
            encoded[i] = bytes1(uint8((_len / (256**(lenLen - i))) % 256));
        }
    }

    return encoded;
}
```


7. Basic coding principles were not followed

Location	Contract file	Risk Status	Risk level
Line108-Line124	L1CrossLayerWitness.sol	 Acknowledged	Minor

① Description

The four functions "blockMessage", "allowMessage", "pause" and "unpause" require `require(msg.sender == address(addressResolver.dao()), "only dao allowed");` . `addressResolver.dao()` in the AddressManager contract implements the `IDAO` interface. However, there is no corresponding function call function in the `DAO` contract.。

② Recommendation

It is suggested to add the calling functions of blockMessage function, allowMessage function, pause function and unpause function respectively at the corresponding positions in the Dao contract.

③ Code

Solidity/**


```
function blockMessage(bytes32[] memory _messageHashes) public {  
  
    //@OKLink Audit Description: The Dao contract does not call this function.  
    //@OKLink Audit Solution: Add the function of blockMessage call in Dao contract.  
  
    require(msg.sender == address(addressResolver.dao()), "only dao allowed");  
    for (uint256 i = 0; i < _messageHashes.length; i++) {  
        blockedMessages[_messageHashes[i]] = true;  
    }  
    emit MessageBlocked(_messageHashes);  
}
```

```
function allowMessage(bytes32[] memory _messageHashes) public {  
  
    //@OKLink Audit Description: The Dao contract does not call this function.  
    //@OKLink Audit Solution: Add the function of allowMessage call in Dao contract.  
  
    require(msg.sender == address(addressResolver.dao()), "only dao allowed");  
    for (uint256 i = 0; i < _messageHashes.length; i++) {  
        blockedMessages[_messageHashes[i]] = false;  
    }  
    emit MessageAllowed(_messageHashes);  
}
```

```
function pause() public {  
  
    //@OKLink Audit Description: The Dao contract does not call this function.  
    //@OKLink Audit Solution: Add the function of pause call in Dao contract.  
  
    require(msg.sender == address(addressResolver.dao()), "only dao allowed");  
    _pause();  
}
```

```
function unpause() public {  
  
    //@OKLink Audit Description: The Dao contract does not call this function.  
    //@OKLink Audit Solution: Add the function of unpause call in Dao contract.  
  
    require(msg.sender == address(addressResolver.dao()), "only dao allowed");  
    _unpause();  
}
```

8. Basic coding principles were not followed

Location	Contract file	Risk Status	Risk level
Line26	L2CrossLayerWitness.sol	 Acknowledged	Minor

① Description

The relayMessage has the same function as the replayMessage, but Merkle verification will not be performed when the caller in the function relayMessage is "constants. L1_cross_layer_witness". Since the "constants. L1_cross_layer_witness address" is unknown, it is impossible to know whether the address has verified the "relayMessage" in advance, but Merkle verification is not performed in the function relayMessage, so there is a security risk.

② Recommendation

It is recommended to launch a verification of Merkle.

③ Code

Solidity/**

```
function relayMessage(
    address _target,
    address _sender,
    bytes memory _message,
    uint64 _messageIndex,
    bytes32 _mmrRoot,
    uint64 _mmrSize
) public returns (bool) {
    require(crossLayerMsgSender == address(0), "reentrancy");
    require(msg.sender == Constants.L1_CROSS_LAYER_WITNESS, "wrong sender");
    bytes32 _hash = CrossLayerCodec.crossLayerMessageHash(_target, _sender,
_messageIndex, _message);

    //@OKLink Audit Description: Merkle verification is not performed
    //@OKLink Audit Solution: Merkle verification is recommended

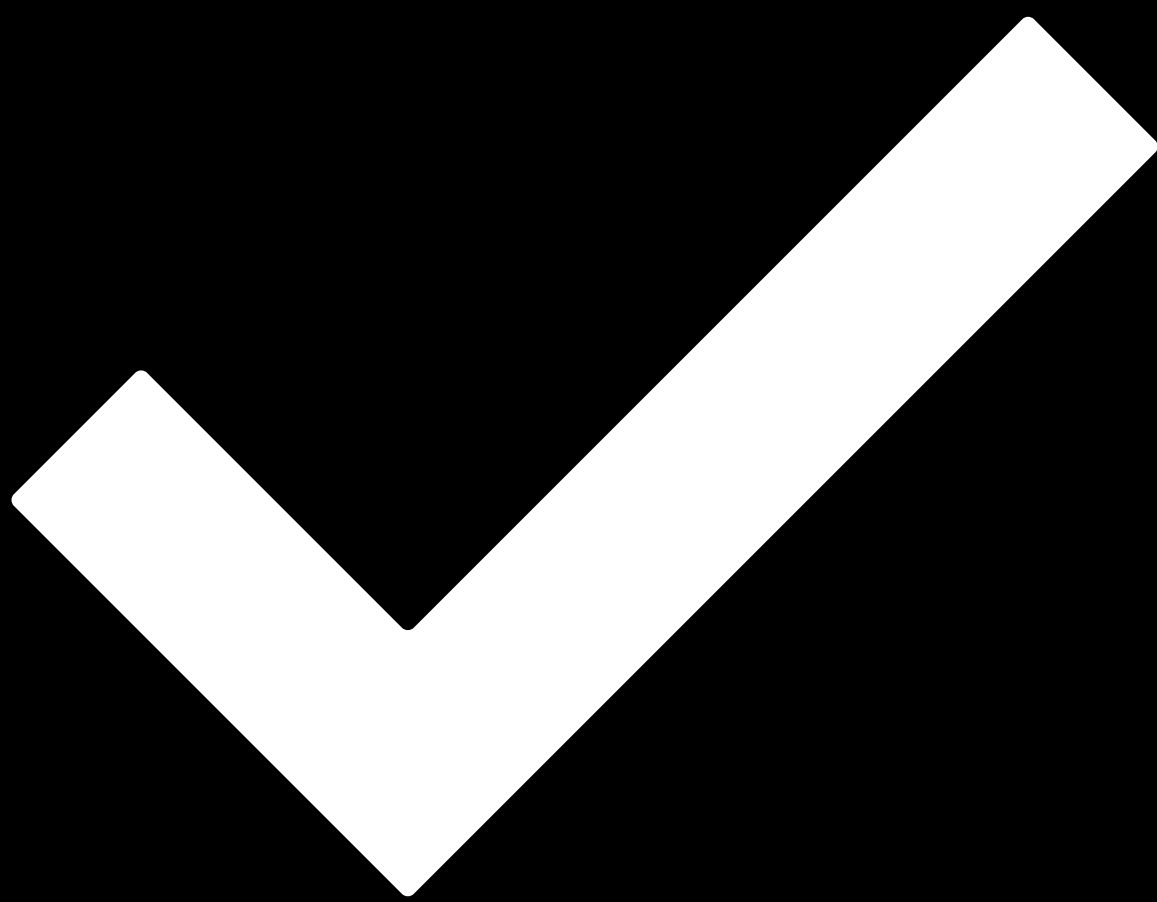
    require(successRelayedMessages[_hash] == false, "already relayed");
    crossLayerMsgSender = _sender;
    (bool success, ) = _target.call(_message);
    crossLayerMsgSender = address(0);
    if (success) {
        successRelayedMessages[_hash] = true;
        emit MessageRelayed(_messageIndex, _hash);
    } else {
        mmrRoots[_mmrSize] = _mmrRoot;
        emit MessageRelayFailed(_hash, _mmrSize, _mmrRoot);
    }
    return success;
}

function replayMessage(
    address _target,
    address _sender,
    bytes memory _message,
    uint64 _messageIndex,
    bytes32[] memory _proof,
    uint64 _mmrSize
) public returns (bool) {
    require(crossLayerMsgSender == address(0), "reentrancy");
    bytes32 _hash = CrossLayerCodec.crossLayerMessageHash(_target, _sender,
_messageIndex, _message);
    bytes32 _mmrRoot = mmrRoots[_mmrSize];
    require(_mmrRoot != bytes32(0), "unknown mmr root");
    MerkleMountainRange.verifyLeafHashInclusion(_hash, _messageIndex, _proof, _mmrRoot,
_mmrSize);
    require(successRelayedMessages[_hash] == false, "message already relayed");
    crossLayerMsgSender = _sender;
    (bool success, ) = _target.call(_message);
    crossLayerMsgSender = address(0);
    if (success) {
        successRelayedMessages[_hash] = true;
        emit MessageRelayed(_messageIndex, _hash);
    } else {
        emit MessageRelayFailed(_hash, _mmrSize, _mmrRoot);
    }
    return success;
}
```


Disclaimer

- i. This audit report focuses only on the types of audits identified in the final report issued. Other unknown security vulnerabilities are not part of this audit, and we do not accept responsibility for them.
- ii. We shall only issue an audit report based on an attack or vulnerability that existed or occurred before the issuance of the audit report. We cannot determine the likely impact on the security posture of our projects for new attacks or vulnerabilities that may exist or occur in the future, and we are not responsible for them.
- iii. The security audit analysis and other elements of our published audit report shall be based solely on documents and materials (including, but not limited to, contract codes) provided to us by the Project Party before the release of the audit report. Such documents and materials shall not be untrue, inaccurate, uninformative, altered, deleted, or concealed, and if the documents and materials provided by the Project Party are false, inaccurate, uninformative, changed, deleted or hidden, or if the documents and materials provided by the Project Party are untrue, inaccurate, uninformative, altered, deleted or concealed, or if the documents and materials provided by the Project Party are uninformative, uninformative, altered, deleted or hidden. If the records and information provided by the Project Party are untrue, inaccurate, uninformative, altered, deleted, or concealed, or if changes to such documents and information are made after the issuance of the audit report, we shall not be liable for any loss or adverse effect arising from any inconsistency between the reflected and actual conditions.
- iv. The Project Parties are aware that our audit report is based on documents and information provided by the Project Parties and relies on the technology currently available. However, due to the technical limitations of any organization, there is a possibility that our audit report may not fully detect all risks. Our audit team encourages the project development team and any interested parties to conduct subsequent testing and audits of the project.
- v. The project owner warrants that the project for which we are engaged to provide audit or testing services is legal, compliant, and does not violate applicable laws. The audit report is for the project owner's reference only, and the contents, manner of obtaining, use of, and any services or resources involved in the audit report shall not be relied upon for investment, tax, legal, regulatory, or advisory purposes of any kind, and we shall not be liable therefor. The Project Party shall not refer to, quote, display, or send the Audit Report in whole or in part to any third party without our prior written consent. The Project Party shall bear any loss or liability arising from that place. We assume no responsibility for any reliance on or use of the audit report for any purpose.
- vi. This audit report does not cover the compiler of the contract or any areas beyond the programming language of the Smart Contract. The risk and liability of the audited Smart Contract arising from references to off-chain information or resources is the sole responsibility of the project party.

- vii. Force Majeure. Force majeure means an unforeseen event whose occurrence and consequences cannot be avoided and cannot be overcome by the parties at the time of entering into the contract, including but not limited to natural disasters such as war, typhoon, flood, fire, earthquake, tidal wave, lightning, natural disaster, strike, nuclear explosion, epidemic and other unforeseen events such as changes in laws, regulations and policies and governmental acts, whose occurrence and consequences cannot be prevented or avoided, and which contains, affects or delays the performance by either party of all or part of its obligations under the contract.
- viii. Suppose either party believes that the occurrence of force majeure affects the performance of its obligations under this Agreement. In that case, it shall promptly notify the other party and, depending on the extent of the effect of the event on the performance of the Agreement; the parties shall consult to determine whether to terminate the Agreement or partially relieve itself of its obligations to perform the Agreement, or to extend the performance of the Agreement.
- ix. In force majeure, neither party shall be deemed in breach or non-performance of its obligations under this Agreement. Any financial commitments existing before the event shall not be affected, and the project party shall make payment for work performed by us.



Passed.

Date 23rd August 2022

Audit Team 歐科雲鏈

The purpose of this audit is to review the layer 2 Rollup smart contract written by Goshen Network based on the solidity, study its design and architecture, and try to find possible vulnerabilities and potential security risks.