Goshen Network

合約審計報告

VER 1.0

2022年8月22日

No. 2022082219020

項目總結

1. 項目介紹

Goshen Network智慧合約主要實現了layer 2交易批量提交，layer 1與layer 2跨層通訊，狀態機，以及窗口期狀態挑戰等功能，為optimistic rollup機制，實現了將交易經過匯總打包後批量上傳到Layer1主鏈，確保任何人都能够通過同步layer1的區塊來完整推導出layer2的狀態。 單個Rollup區塊中可以打包大量交易，由此可以有效提升主鏈的輸送量。

2. 審計詳情

項目名稱	Goshen Network	平台	N/A
通證名稱	N/A	通證代號	N/A
開始時間	2022年7月26日	語言	Solidity
結束時間	2022年8月19日	官網	N/A
Github	https://github.com/ontology-layer-2/rollup-contracts/tree/e3634932ba55ffc256d72180306f091acdbd1fc4/contracts	白皮書	N/A

3. 審計範圍

ID	文件	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	b43d3ff99f0f1041325517049 34ce33c72797395809026d5bc9aa3756738aaf3
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	Interpretor.t.sol	7bda315fa3d2bea5a0f18ed74dfac0d614263685f3276b179bda728653379d88
contracts/state-machine/riscv32	Interpretor.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	98e41a5fcb84ac0256eab1354724c6b500e43d3bfb6f9786463be98c84248b4e
contracts/bridge	L1StandardBridge.sol	41899439179b6b24f04f4a06e5ddeb0420c38a9f7e3755bfb902eaaaa9fcf854
contracts/bridge	L2StandardBridge.t.sol	b8f1c37721222c6aba38a9ab96191f8af74de916d72d49f1a72224bdd099c2ff
contracts/bridge	L2StandardBridge.sol	6d262e6f8aad79e4f8451d39bf3501d03ade150e262953cc36938b4d20f232cd

ID	文件	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	文件	SHA-256 checksum
contracts/rollup	RollupStateChain.t.sol	c70d0fbebf7723996d039f89e5080b3e36daddbce5f5ce47abd0972fd4ac412d
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	3631db453463545f711831783b50b546d4ee9e424466e333ffcf09f8b140ecf3
contracts/challenge	DisputeTree.sol	d4049a4a5d26d1c11554a799845a52fecd8328b32f590b6bc7c717b97c8a6e38
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	文件	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	9f26c185bfe020370dcb45d3e1dc344f6265af1185f0ee1c3258b4b402fdcb32
contracts/interfaces	IChallengeFactory.sol	a7f88515fb211d42525ed7862eca97c9971778ac6a7c8a5890c41e48e0388a68
contracts/interfaces	IL1CrossLayerWitness.sol	6ff29d2b7dc0a56a3a56a89ea0ba33e7957536d61163bf302916dde4ea376325
contracts/interfaces	ICrossLayerWitness.sol	f16d6cbcc5b39e73edc170a27839872118822525 30b1161feaa10f4f0dc36a92
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. 代碼結構

```
contracts
├── bridge                                  #標準橋實現
|   ├── L1StandardBridge.sol
|   ├── L1StandardBridge.t.sol
|   ├── L2StandardBridge.sol
|   └── L2StandardBridge.t.sol
├── builtins
|   ├── L2FeeCollector.sol
|   └── L2FeeCollector.t.sol
├── challenge                               #欺詐挑戰
|   ├── Challenge.sol
|   ├── ChallengeFactory.sol
|   ├── ChallengeFactory.t.sol
|   ├── DisputeTree.sol
|   └── DisputeTree.t.sol
├── cross-layer                             #層間通訊
|   ├── CrossLayerCodec.sol
|   ├── CrossLayerContext.sol
|   ├── CrossLayerWitness.t.sol
|   ├── L1CrossLayerWitness.sol
|   ├── L1CrossLayerWitness.t.sol
|   └── L2CrossLayerWitness.sol
├── dao
|   └── Whitelist.sol
├── interfaces                              #介面檔案
|   ├── ForgeVM.sol
|   ├── IAddressManager.sol
|   ├── IAddressResolver.sol
|   ├── IChainStorageContainer.sol
|   ├── IChallenge.sol
|   ├── IChallengeFactory.sol
|   ├── ICrossLayerWitness.sol
|   ├── IInterpretor.sol
|   ├── IL1CrossLayerWitness.sol
|   ├── IL1ERC20Bridge.sol
|   ├── IL1StandardBridge.sol
|   ├── IL2CrossLayerWitness.sol
|   ├── IL2ERC20Bridge.sol
```

```
contracts
|   ├── IL2StandardERC20.sol
|   ├── IMachineState.sol
|   ├── IRollupInputChain.sol
|   ├── IRollupStateChain.sol
|   ├── IStakingManager.sol
|   ├── IStateTransition.sol
|   └── IWhitelist.sol
├── libraries                                  #庫檔案
|   ├── 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                                     #rollup chain和state打包
|   ├── ChainStorageContainer.sol
|   ├── ChainStorageContainer.t.sol
|   ├── RollupInputChain.sol
|   ├── RollupInputChain.t.sol
|   ├── RollupStateChain.sol
|   └── RollupStateChain.t.sol
```

```
contracts
├── staking
│   ├── StakingManager.sol
│   └── StakingManager.t.sol
├── state-machine                    #狀態機實現
│   ├── 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
```

審計報告匯總

1. 審計方式

通過清晰地理解該項目的設計目的、運行原理和實現管道，稽核團隊對合約程式碼進行了深入的研究和分析。 在分清各個合約及其函數的調用關係的基礎上，對合約可能存在的漏洞進行了定位及分析。 最終產生問題描述和給出相應的修改意見。

審計方法	Static analysis, Manual Review

2. 審計流程

步驟	操作	詳細內容
1	背景研究	閱讀項目介紹、白皮書、合約源碼等項目方團隊提供的相關信息，確保正確理解項目功能
2	自動化檢測	主要用自動化工具掃描源碼，找到常見的潛在漏洞
3	人工審閱	工程師逐行閱讀代碼，找到潛在漏洞
4	邏輯校對	工程師將對代碼的理解和項目方提供的信息比較，檢查代碼實現是否符合項目白皮書信息
5	測試用例檢測	包括測試用例設計，測試範圍分析、符號執行等
6	優化審查	根據應用場景、調用方式及最新的研究成果從可維護性、安全性及可操作性等方面審查項目

3. 風險分級

風險級別	風險描述
致命	存在致命風險及隱患，需要立即解決
高危	存在高危風險及隱患，將引發相同問題，必須解決
中度	存在中度風險及隱患，可能導致潛在風險，最終仍然需要解決
低危	存在低風險及隱患，指各類處理不當或會引發警告信息的細節，這類問題可暫時擱置
建議	存在可優化的部分，這類問題可以擱置，但建議最終解決

4. 審計結果

編號	審計項目	風險級別	狀態
1	重入	無	
2	注入	無	
3	權限繞過	無	
4	Mempool搶跑	無	
5	回滾	無	
6	條件競爭	無	
7	循環耗盡gas	無	
8	閃電貸高影響	無	
9	經濟模型不合理	無	
10	可預見的隨機數	無	
11	投票權管理混亂	無	

編號	審計項目	風險級別	狀態
12	數據隱私洩露	無	
13	鏈上時間使用不當	無	
14	Fallback函數編碼不當	無	
15	許可權處理不當	高	已告知
16	內寘函數使用不當	無	
17	內聯匯編使用不當	無	
18	構造函數不規範	無	
19	返回值不規範	無	
20	Event不規範	無	
21	關鍵字使用不規範	無	
22	未遵循ERC標準	無	
23	條件判斷不規範	高	已告知
24	流動性枯竭風險	無	
25	中心化風險	無	
26	邏輯變更風險	無	
27	整數溢出	無	
28	函數可見性不當	無	
29	變量初始化不當	無	
30	合約間調用不當	無	
31	變量不規範	無	
32	重放	無	
33	隨機存儲位置寫入	無	
34	蜜罐邏輯	無	
35	哈希碰撞	無	
36	領獎邏輯不當	無	
37	使用不推薦的方法	無	
38	未遵循基本編碼原則	低	已告知
39	過時的外部依賴	無	
40	業務邏輯存在問題	中	已告知

上述表格中，狀態欄內容若為「已告知」，則表示審計團隊已告知項目方項目存在的漏洞，但項目方未對漏洞進行修改，或未告知審計團隊漏洞的修改進度。若狀態欄中填寫「已修改」則表示項目方已進行對漏洞的修改，並通過審計團隊確認。

5. 風險項與修改方案

以下部分為審計後得知的風險項相關詳細信息，其中內容包括風險類型、風險級別、問題位置、問題描述、修改建議及項目方反饋。

1. 業務邏輯存在問題

位置	文件	風險状态	風險級別
Line77-Line80，Line88-Line110	StakingManager.sol	⚠️已告知	中風險

① 風險描述

Proposer被驗證作惡以後，`getStakingInfo[proposer]` 中除了StakingState被設為UNSTAKED以外，其他的資訊沒有改變（Line88-Line110）。 Proposer可以再次通過deposit把自己帳戶的StakingState設為STAKING，然後再次作惡，此時challenger對其進行挑戰時，將無法通過require檢查（Line 77-80）`proposerStake. earliestChallengeHeight == 0 || _ chainHeight < proposerStake. earliestChallengeHeight` ，無法進行挑戰。

② 修改建議

將proposerStake中的所有參數進行重置。

③ 關聯程式碼

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: 只重置了state，沒有重置其他資訊。
    //@OKLink Audit Solution: 將proposerStake中的所有參數進行重置。
---------------------------------------------------------------------------------
    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: 只重置了state，沒有重置其他資訊。
    //@OKLink Audit Solution: 將proposerStake中的所有參數進行重置。
---------------------------------------------------------------------------------
    proposerStake.state = StakingState.UNSTAKED;
    emit DepositClaimed(_proposer, DAOAddress, price);
}
```

2. 條件判斷不規範

位置	文件	風險状态	風險級別
Line40	ChallengeFactory.sol	⚠️已告知	高風險

① 風險描述

在newChallange函數中，require語句判斷條件錯誤，導致無法新建challenge。

，無法進行挑戰。

② 修改建議

建議將程式碼行require(challengedStates[_hash] != address(0), "already challenged")

修改為require(challengedStates[_hash] == address(0), "already challenged")

③ 關聯程式碼

Solidity

```
/**
function newChallange(
        //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: 新建challenge時，challengedStates[_hash]為0值。
        //@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. 許可權處理不當

位置	文件	風險状态	風險級別
Line55	ChallengeFactory.sol	⚠️已告知	高風險

① 風險描述

Challenger進行欺詐挑戰時，通過ChallengeFactory合約newChallange介面，創建新挑戰，並調用Challenge合約的create函數，此時挑戰創建者需要deposit一定的token，並通過transferFrom從creator帳號充值到對應Challenge的BeaconProxy合約地址。 由於BeaconProxy合約為函數newChallange調用過程中產生，並立即發生transferFrom調用，該過程中間未對BeaconProxy合約進行approve，導致transferFrom失敗。

② 修改建議

建議調用approve進行授權，允許BeaconProxy合約轉移用戶token

③ 關聯程式碼

Solidity

```
/**
    function newChallange(
        //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: BeaconProxy合約為當前函數創建，未進行approve即調用
transferFrom
        //@OKLink Audit Solution: 調用approve進行授權，允許BeaconProxy合約轉移用戶token
```

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

4. 業務邏輯存在問題

位置	文件	風險状态	風險級別
Line260-Line275	Challenge.sol	⚠️已告知	中風險

① 風險描述

Challenger進行欺詐挑戰成功後，如果有多個challenger參與，需按照樹根自上而下依次權重遞增分配獎勵。 該獎勵分配算濃程式碼實現與注釋描述不相符，且需考慮該獎勵分配算濃加和是否能分配完獎勵資金，如果未能分配完獎勵資金，需新增對剩餘獎勵資金的操作函數。

② 修改建議

建議修改獎勵分配算濃實現

③ 關聯程式碼

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: 獎勵分配實現算濾與注釋不符
        //@OKLink Audit Solution: 修改獎勵分配算濾實現
```

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

5. 業務邏輯存在問題

位置	文件	風險状态	風險級別
Line69-Line90	Challenge.sol	⚠️已告知	中風險

① 風險描述

任意用戶可以多次調用Chanllenge合約的create函數。 Challenge合約沒有限制create函數只能够調用一次，這使得Challenge合約的狀態可以被任意外部調用進行重置。 在approve額度足够的前提下，可以把原始創建者的token轉進Challenge合約。

② 修改建議

建議增添一個stage判斷：require(stage == ChallengeStage.Uninitialized, "only uninitialized stage");

③ 關聯程式碼

Solidity

```
/**
function create(
        bytes32 _systemStartState,
        address _creator,
        uint256 _proposerTimeLimit,
        Types.StateInfo memory _stateInfo,
        uint256 _minChallengerDeposit
    ) external override {
---------------------------------------------------------------------------------------------------
        //@OKLink Audit Description: 沒有對當前合約的狀態進行判斷
        //@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. 業務邏輯存在問題

位置	文件	風險状态	風險級別
Line95	RLPWriter.sol	⚠️已告知	中風險

① 風險描述

`writelength` 函數中，`encoded[0] = bytes1 (uint8 (lenLen + _offset + 55) )` 在進行計算式對結果進行了強制轉換使其溢出，導致在 `writeUint` ，`writeString` ，`writeAddress` 中執行功能時，不同的輸入會導致相同的 `encoded[0]` 輸出

例1：`lenlen=74, _ offset=128` 與 `lenlen=330, _ offset=128` 結果均為 `encoded[0]=0x01`

例2：`lenlen=10, _ offset=192` 與 `lenlen=266, _ offset=192` 結果均為 `encoded[0]=0x01`

在optimism中，計算管道為：`bytes1 (uint8 (_len) + uint8 (_offset) +55)` ；對於不符合要求，超出bytes1範圍限制的參數會返回錯誤。

② 修改建議

建議改為bytes1（uint8（_len）+ uint8（_offset）+55）

③ 關聯程式碼

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: 使用了uint8進行強制轉換
            //@OKLink Audit Solution: 改為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. 未遵循基本編碼原則

位置	文件	風險状态	風險級別
Line108-Line124	L1CrossLayerWitness.sol	⚠️已告知	低風險

① 風險描述

blockMessage, allowMessage, pause, unpause 四个函数要求 `require(msg.sender == address(addressResolver.dao()), "only dao allowed");` ，AddressManager 合约中的 `addressResolver.dao()` 实现的是 `IDAO` 接口。但是 `DAO` 合约中并没有对应的函数调用功能。

② 修改建議

建議在dao合約中相對應的位置分別增添blockMessage函數、allowMessage函數、pause函數、unpause函數的調用功能。

③ 關聯程式碼

Solidity

```
/**
function blockMessage(bytes32[] memory _messageHashes) public {
-----------------------------------------------------------------
    //@OKLink Audit Description: dao合約沒有調用該函數的功能。
    //@OKLink Audit Solution: 在dao合約中添加blockMessage函數調用功能。
-----------------------------------------------------------------
    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: dao合約沒有調用該函數的功能。
    //@OKLink Audit Solution: 在dao合約中添加allowMessage函數調用功能。
-----------------------------------------------------------------
    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: dao合約沒有調用該函數的功能。
    //@OKLink Audit Solution: 在dao合約中添加pause函數調用功能。
-----------------------------------------------------------------
    require(msg.sender == address(addressResolver.dao()), "only dao allowed");
    _pause();
}

function unpause() public {
-----------------------------------------------------------------
    //@OKLink Audit Description: dao合約沒有調用該函數的功能。
    //@OKLink Audit Solution: 在dao合約中添加unpause函數調用功能。
-----------------------------------------------------------------
    require(msg.sender == address(addressResolver.dao()), "only dao allowed");
    _unpause();
}
```

8. 未遵循基本編碼原則

位置	文件	風險状态	風險級別
Line26	L2CrossLayerWitness.sol	⚠️已告知	低風險

① 風險描述

函數relayMessage與函數replayMessage功能相同，但函數relayMessage中調用者為Constants.L1_CROSS_ LAYER_ WITNESS時並不會進行Merkle校驗。 由於對Constants. L1_ CROSS_ LAYER_ WITNESS地址未知，並不能知道該地址對發送的relayMessage是否進行提前驗證，但是在函數relayMessage中也未進行Merkle校驗，存在安全風險。

② 修改建議

建議進行Merkle校驗

③ 關聯程式碼

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: 未做Merkle校驗
        //@OKLink Audit Solution: 推薦進行Merkle校驗
```

```
        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 {
            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;
    }
```

免責聲明

i. 本審計報告僅針對最終出具報告中載明的審計類型進行審計，其他未知安全漏洞不在本次審計責任範圍之內，我方無需為此承擔責任。

ii. 我方僅應根據審計報告發布之前存在或發生的攻擊或漏洞發布審計報告。對於將來存在或發生的新攻擊或漏洞，我方無法確定對其項目安全狀態的可能影響，對此概不負責。

iii. 我方發布的審計報告中的安全審計分析及其他內容應僅基於項目方在發布審計報告之前向我方提供的文件和材料（包括但不限於合約代碼），並且上述文件和資料不應該存在缺乏信息、被篡改、刪除或隱藏的情況，如果項目方提供的文件和資料存在不真實、不準確、缺乏信息、被篡改、刪除或隱藏的情況，或者對上述文件和資料的改動是在發布審計報告之後作出的，我方不承擔因反映情況與實際情況不一致引起的損失和不利影響 。

iv. 項目方知曉我方出具的審計報告系根據項目方提供的文件和資料、依靠我方現掌握的技術而作出的。但由於任何機構均存在技術的局限性，我方作出的審計報告仍存在無法完整檢測出全部風險的可能性。我方審計團隊鼓勵項目的開發團隊以及任何相關利益方對項目進行後續的測試及審計。

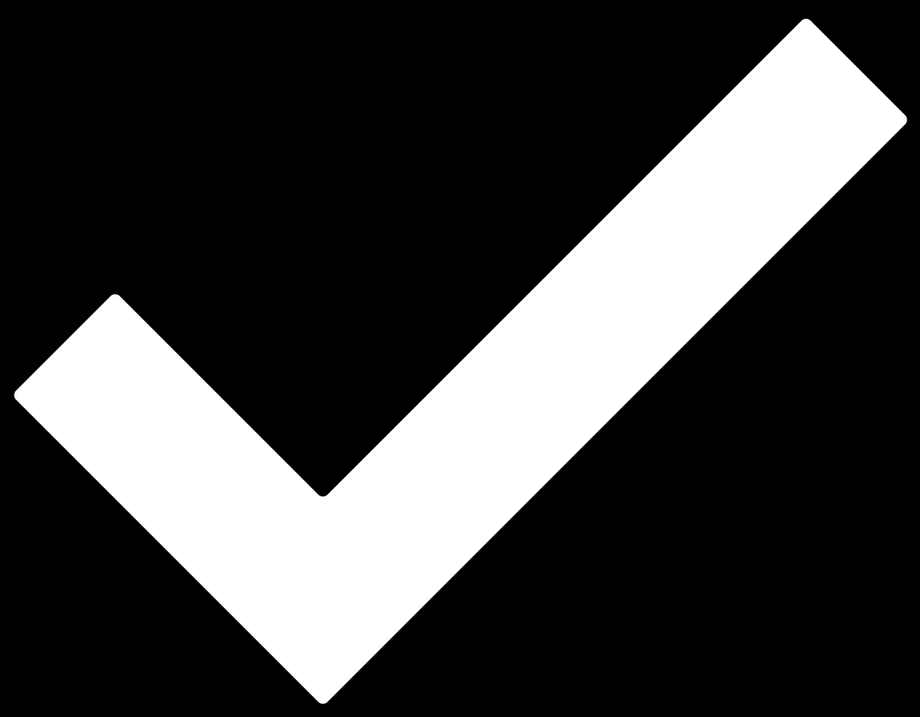
v. 項目方保證其委托我方提供審計或測試服務的項目合法、合規，且不違反適用法律。審計報告僅用於項目方參考，審計報告的內容、獲取方式、使用以及任何其所涉及的服務或資源都不能作為任何形式的投資、稅務、法律、監管及建議等的依據，我方不因此承擔相關責任。在未經我方書面同意之前，項目方不得將審計報告的全部或部分內容以任何形式提及、引用、展示或發送給任何第三方，否則由此產生的任何損失和責任由項目方自行承擔。我方對任何人依賴審計報告或將之用於任何目的概不承擔責任 。

vi. 本審計報告不涉及合約的編譯器及任何超出智能合約編程語言的領域，所審計的智能合約因引用鏈下信息或資源所導致的風險及責任，由項目方自行承擔 。

vii. 不可抗力。不可抗力是指雙方在訂立合同時不能預見、對其發生和後果不能避免且不能克服的事件，包括但不限於戰爭、臺風、水災、火災、地震、潮汐、雷電、天災、罷工、核爆炸、流行病等自然災害和法律、法規和政策變更及政府行為等其它不可預見，對其發生和後果不能防止或避免的事件，且該事件妨礙、影響或延誤任何一方根據合同履行其全部或部分義務。

viii. 如果有一方認為不可抗力發生影響履行本協議義務，應迅速通知另一方，按事件對履約影響的程度，由雙方協商決定是否終止合同或部分免除履約的責任，或者延期履約。

ix. 當不可抗力發生時，任何一方都不能被視作違約或不履行本協議義務。在事件前存在的經濟上的責任，不應受到影響，項目方應對我方已完成工作做出支付。



審計通過.

日期 2022年8月22日

審計 歐科雲鏈

本次稽核的目的是為了審閱Goshen Network項目基於Solidity語言編寫的Layer 2 Rollup智慧合約，研究其設計、架構，發現潛在的安全隱患，並試圖找到可能存在的漏洞。