MAP Protocol

合約審計報告

VER 1.0

2022年7月11日

No. 2022071110590

項目總結

1. 項目介紹

MAP Protocol通過構建全鏈網絡基礎設施，提升整個跨鏈生態系統。 任何使用MAP Protocol的開發者都會自動繼承全鏈性質，而無需開發者處理容易出錯的跨鏈通信，從而降低建設全鏈dApp的科技門檻。

1.MAP Protocol分為三層：協定層、跨鏈服務層（MCS）和應用層。 2.MAP Protocol通過鏈上智慧合約進行驗證。 3.MAP Protocol的跨鏈驗證網絡採用輕用戶端獨立自驗證

2. 審計詳情

項目名稱	MAP Protocol	平台	N/A
通證名稱	N/A	通證代號	N/A
開始時間	2022年6月29日	語言	Solidity
結束時間	2022年7月11日	官網	N/A
Github	https://github.com/mapprotocol/map-contracts/tree/ea53c8931353c6693bdcdd0c996a3a9319d784f8/mapclients/eth/contracts	白皮書	N/A

2022年7月11日

3. 審計範圍

ID	文件	SHA-256 checksum
contracts	LightNode.sol	a4b29ddf406554d0c6b798e773048a04e50dad8774a1cc35985f7d963df18319
contracts	LightNodeProxy.sol	1a265a3c24a9ddba8f236f4abb4f7f9e7612a0535024d52a140f295152e48610
contracts	VerifyTool.sol	7d12b26af32038cc9c8a78a4f8a84d88b5700d1e09f9a72e9810fd4bc350c3b3
contracts/lib	MPT.sol	3ab7fb3466ca64f234964fd26daaa507d45700fa193b8adcc1b1362435b83cea
contracts/lib	RLPEncode.sol	cd552d56e3589eeff4fb4286e7c4527630808aa16a817ff903b0fae012a70181
contracts/lib	RLPReader.sol	b06774b7ab7ed057fca9b76f0d719b826d616c8e08ccb5e41e1a8c175a9571b0
contracts/bls	BGLS.sol	fa630679679afad4edc784521df7d25a6f3c391aaa1bf9e895b541be19c5b271
contracts/bls	BlsCode.sol	62196c5e1f4e67833d24f3a3a288ef7bae0f9f565e06fdfba25a0180cc256579

4. 代碼結構

```
├── LightNode.sol                    #輕節點驗證邏輯
├── LightNodeProxy.sol               #ERC1967代理合約
├── VerifyTool.sol                   #MPT、簽名驗證工具合約
├── bls
|   ├── BGLS.sol                     #BGLS聚合簽名
|   └── BlsCode.sol                  #BLS編解碼
├── interface                        #介面定義
|   ├── ILightNode.sol
|   ├── ILightNodePoint.sol
|   └── IVerifyTool.sol
└── lib
    ├── MPT.sol                      #MPT驗證lib
    ├── RLPEncode.sol                #RLP編碼lib
    └── RLPReader.sol                #RLP讀取器lib
```

2022年7月11日

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

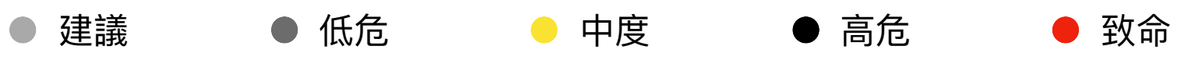
步驟	操作	詳細內容
1	背景研究	閱讀項目介紹、白皮書、合約源碼等項目方團隊提供的相關信息，確保正確理解項目功能
2	自動化檢測	主要用自動化工具掃描源碼，找到常見的潛在漏洞
3	人工審閱	工程師逐行閱讀代碼，找到潛在漏洞
4	邏輯校對	工程師將對代碼的理解和項目方提供的信息比較，檢查代碼實現是否符合項目白皮書信息
5	測試用例檢測	包括測試用例設計，測試範圍分析、符號執行等
6	優化審查	根據應用場景、調用方式及最新的研究成果從可維護性、安全性及可操作性等方面審查項目

2022年7月11日

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	

2022年7月11日

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	鑒權不當	無	
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	無	
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	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。

2022年7月11日

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.未遵循基本編碼原則

位置	文件	風險状态	風險級別
Line 8	LightNode.sol	⚠ 已告知	中風險

① 風險描述

import不存在的檔名，會導致編譯不通過。

② 修改建議

建議如無特別計畫，去掉不存在的import。

2022年7月11日

③ 关联代码

JavaScript/**

```
import "@openzeppelin/contracts/proxy/utils/UUPSUpgradeable.sol";
import "@openzeppelin/contracts/proxy/utils/Initializable.sol";
import "./interface/ILightNode.sol";
//@OKLink Audit Description: import未創建的檔案
//@OKLink Audit Solution: 去掉該語句
import "./interface/IWeightedMultiSig.sol";
import "./bls/BlsCode.sol";
import "./bls/BGLS.sol";
import "./interface/IVerifyTool.sol";
```

2022年7月11日

2. 未遵循基本編碼原則

位置	文件	風險状态	風險級別
Line 16~18	LightNode.sol	⚠ 已告知	中風險

① 風險描述

常數未使用constant關鍵字。 在solidity中，聲明為constant或者immutable的狀態變數即常數。 constant修飾的常數的值在編譯時確定，而immutable修飾的常數的值在部署時確定。 儘量使用常數，常數是合約位元組碼的一部分，不佔用存儲插槽，使用常數比變數更省gas。 在部署時，常數消耗的gas更少。

② 修改建議

建議將這三行狀態變數關鍵字修改為constant

2022年7月11日

③ 关联代码

```
JavaScript/**
     * contract LightNode is UUPSUpgradeable,Initializable, ILightNode,BGLS {

//@OKLink Audit Description: 未使用constnt

//@OKLink Audit Solution: 使用constant關鍵字標識常數

    uint256 public maxValidators = 20;

    uint256 public epochSize = 1000;

    uint256 public headerHeight = 0;

    address[] public validatorAddresss;

    validator[maxValidators] public validators;
```

2022年7月11日

3. 未遵循基本編碼原則

位置	文件	風險状态	風險級別
Line 80	LightNode.sol	⚠ 已告知	中風險

① 風險描述

程式碼中如果存在定義的變數未使用，或賦值後未使用的情況，可能會導致編碼時的邏輯錯誤，或平白消耗gas。 清除未使用的變數是編碼的良好規範，也有助於提升程式碼可讀性和可維護性。
此處`hash`變數定義後，通過`verifyTool. verifyHeader（headerRlp）`完成了賦值，但後續在未有過對`hash`的調用。

② 修改建議

建議檢查程式碼邏輯是否正確，如果確實無用，建議清除。

2022年7月11日

③ 关联代码

```
JavaScript/**
function verifyProofData(receiptProof memory _receiptProof)
    external
    view
    override
    returns (bool success, string memory message,bytes memory logsHash) {
        logsHash = verifyTool.encodeTxLog(_receiptProof.receipt.logs);
        (success, message) =verifyTool.getVerifyTrieProof(_receiptProof);
        if (!success) {
            message = "receipt mismatch";
            return (success, message,logsHash);
        }
        //@OKLink Audit Description: 沒有使用到hash這個局部變數，後面hash賦值後再未
使用
        //@OKLink Audit Solution: 建議檢查清楚邏輯是否正確，如確實無用，建議删除
        bytes32 hash;
        bytes memory headerRlp =
verifyTool.encodeHeader(_receiptProof.header);
        (success, hash) = verifyTool.verifyHeader(headerRlp);
        if (!success) {
            message = "verifyHeader error";
            return (success, message,logsHash);
        }
        istanbulExtra memory ist =
verifyTool.decodeExtraData(_receiptProof.header.extraData);
        success = checkSig(_receiptProof.header, ist, _receiptProof.aggPk);
        if (!success) {
            message = "bls error";
        }
        return (success, message,logsHash);
    }
```

2022年7月11日

4. 未遵循基本編碼原則

位置	文件	風險状态	風險級別
Line 22	LightNode.sol	⚠ 已告知	中風險

① 風險描述

狀態變數初始化的規範是放在初始化函數或constructor函數中去賦值，以避免有可能的重複賦值情況。此處在狀態變數定義時直接進行賦值，會造成合約部署時過量的gas消耗，也會導致後續可能出現的重複初始化。 同時此處直接new一個合約對象也不符合規範。

② 修改建議

建議使用介面來進行創建或引用，避免gas消耗過多，亦或者，通過lib的管道來使用`BlsCode`的功能（`BlsCode`更像lib合約），而不是每個LightNode合約都要創建一個BlsCode合約。

2022年7月11日

③ 关联代码

```
JavaScript/**

contract LightNode is UUPSUpgradeable,Initializable, ILightNode,BGLS {

    uint256 public maxValidators = 20;

    uint256 public epochSize = 1000;

    uint256 public headerHeight = 0;

    address[] public validatorAddresss;

    validator[maxValidators] public validators;


    IVerifyTool public verifyTool;

    //@OKLink Audit Description: 狀態變數不應該在定義的時候就初始化

    //@OKLink Audit Solution: 在初始化函數中進行賦值，或者繼承，且最好使用地址和
    interface表示

        BlsCode blsCode = new BlsCode();

        ```
```

2022年7月11日
```

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 。

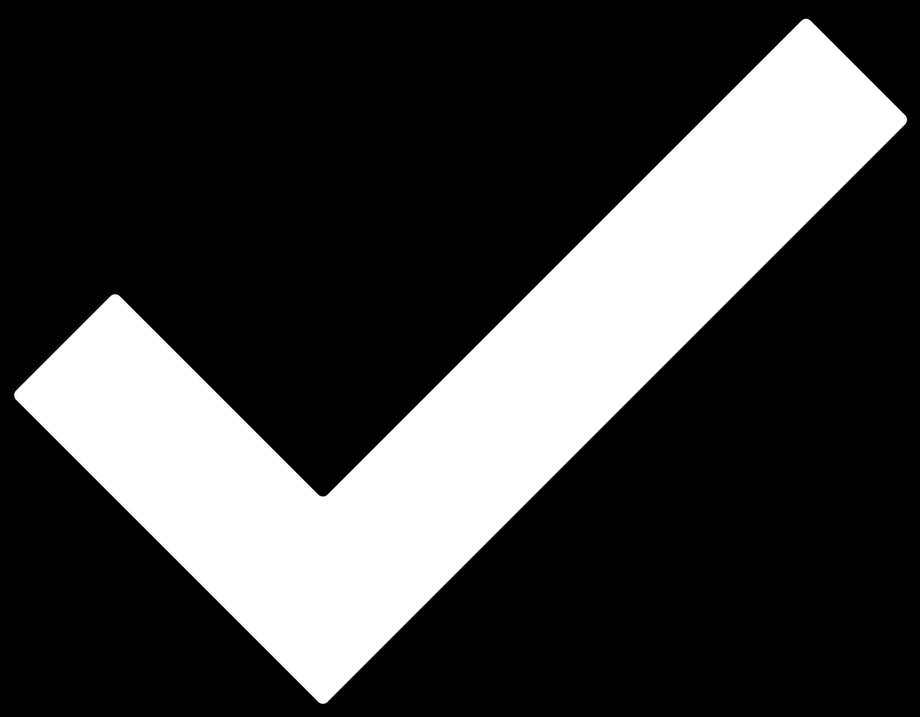
vi. 本審計報告不涉及合約的編譯器及任何超出智能合約編程語言的領域，所審計的智能合約因引用鏈下信息或資源所導致的風險及責任，由項目方自行承擔 。

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。

2022年7月11日



審計通過.

日期 2022年7月11日

審計 歐科雲鏈

本次稽核的目的是為了審閱Map Protocol跨鏈橋項目基於Solidity語言編寫的乙太坊輕節點MPT驗證協定，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。