Ternoa

Bridge Relayer

SMART CONTRACT AUDIT

24.05.2022

Made in Germany by Chainsulting.de

Table of contents

1. Disclaimer

The audit makes no statements or warrantees about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of CAPSULE CORP. LABS SASU. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (11.05.2022)	Layout
0.2 (13.05.2022)	Test Deployment
0.5 (15.05.2022)	Automated Security Testing Manual Security Testing
0.6 (16.05.2022)	Development Productivity (Code Conventions Check, packages)
0.7 (17.05.2022)	Performance (Connection pooling, caching, Transaction Concurrency)
0.8 (17.05.2022)	Maintainability & Ease of Deployment
0.9 (19.05.2022)	Summary and Recommendation
1.0 (19.05.2022)	Final document
1.1 (24.05.2022)	Re-Check a2f81d4e82454cd1d33eb4c12027100f00d66eba

2. About the Project and Company

Company address:

CAPSULE CORP. LABS SASU
19 Bd Malesherbes
75008 Paris
France

Website: https://www.ternoa.com

Twitter: https://twitter.com/ternoa_

GitHub: https://github.com/capsule-corp-ternoa

Telegram: https://t.me/ternoa

Discord: https://discord.gg/cNZTGtGJNR

Twitch: https://www.twitch.tv/ternoatv

Instagram: https://www.instagram.com/ternoa_

Medium: https://medium.com/ternoa

LinkedIn: https://www.linkedin.com/company/ternoa

YouTube: https://www.youtube.com/channel/UCUYvbtRE5HoWPz7z88V7Khw

2.1 Project Overview

Ternoa is an NFT-focused chain based on Substrate that aims to provide the best experience for developers and users of NFT-driven dApps.

Ternoa's unique innovation include the ability for users to store private data within NFTs, user-specified conditions for automated transfers of the secret NFTs to other wallets, lending or sharing NFTs with others while retaining full-rights and NFT ownership, combining multiple NFTs into a new NFT, and sharing of NFT ownership through fractionalization.

Ternoa Bridge

A blockchain bridge provides a connection that allows for the transfer of tokens between two different blockchain ecosystems. A bridge is required for Ternoa because as of the ICO date, CAPS were ERC-20 tokens. However, native CAPS are necessary to take part in staking and rewards on Ternoa, as well as paying transaction fees.

The Ternoa bridge enables CAPS ERC-20 holders to transfer the token onto the Ternoa blockchain, on a 1:1 ratio. Unlike many other bridges, the Ternoa Bridge will lock the ERC20 CAPS and mint the equivalent in native CAPS on the Ternoa Network.

In this manner, if you decide to sell your native CAPS, you will need to transfer them back to the Ethereum network by using the Ternoa Bridge again. ERC20 contracts are built using ChainBridge, a modular Multi-Directional Blockchain Bridge to interact with Ethereum, Substrate, based chains networks.

Documentation: https://docs.ternoa.network/v/bridge/

Frontend: https://bridge.ternoa.com

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the code functioning in a number of scenarios or creates a risk that the code may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a codebase, or provides the opportunity to use an application in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the code in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the code and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pen testers and developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the codebase.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.

2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the codebase to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your codebase.

5. Metrics

The metrics section should give the reader an overview on the size, quality, flows and capabilities of the codebase, without the knowledge to understand the actual code.

5.1 Tested Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
./src/chains/ethereum/ABIs.ts	ec3618813fbc37834032bba59cf4d9d6
./src/chains/ethereum/chain.ts	72baedd1fa5b5ed7c981b561bf2dad42
./src/chains/ethereum/connection.ts	76a87fe0a91c912b749fff02315e3370
./src/chains/ethereum/listener.ts	b19b3a83de87be47a7058fc589a2c261
./src/chains/ethereum/transaction.ts	75659b9003fb53c375a222995907b46a
./src/chains/ethereum/writer.ts	2a8bca4283b9b76331845ba542580b50
./src/chains/substrate/chain.ts	53052c71fb9dbbacb3960dec776c18f5
./src/chains/substrate/connection.ts	55dd463431af7da5034fb209efd4c8c9
./src/chains/substrate/listener.ts	05b0f047fc80bcecc58d46eff1d8c348
./src/chains/substrate/writer.ts	0c1655102db469e95fb39f67894ac2dd
./src/utils/blockstore.ts	45550b8982b8af78c9d050b69e6cdb40
./src/utils/chain-interface.ts	129e3743423b2f466ccf32e452e5172c
./src/utils/core.ts	d4a9702f2f162f409ef9240ead23b101
./src/utils/msg.ts	0d7414023e803ddec820449f8c8b0e57
./src/utils/system.ts	2bd0c7f8efff7048b35391b497ee61d9

5.2 Dependencies

Dependency / Import Path	Source
@badrap/result	https://www.npmjs.com/package/@badrap/result
@polkadot/api	https://www.npmjs.com/package/@polkadot/api
ethers	https://www.npmjs.com/package/ethers

5.3 Codebase Overview

Source: https://github.com/capsule-corp-ternoa/ternoa-bridge-core

language	files	code	comment	blank	total
TypeScript	15	2,251	64	151	2,500

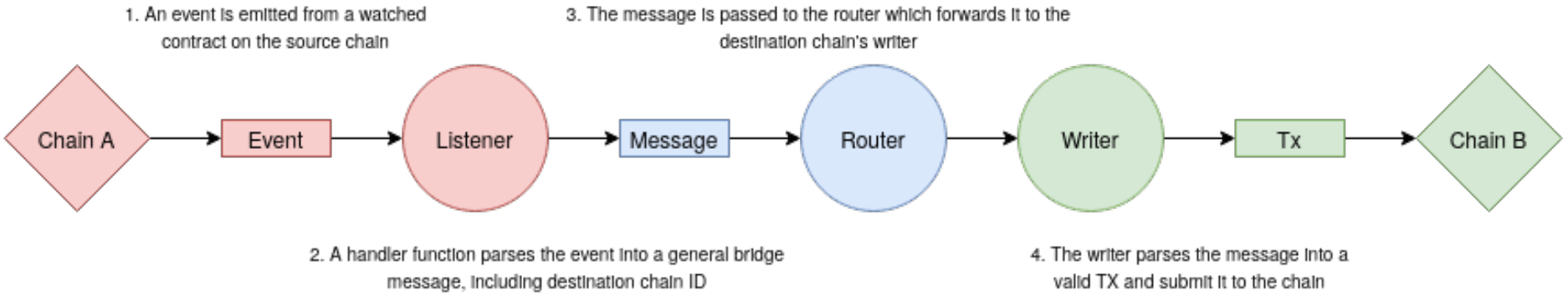
filename	language	code	comment	blank	total
src/chains/ethereum/ABIs.ts	TypeScript	1,233	0	2	1,235
src/chains/ethereum/chain.ts	TypeScript	76	9	18	103

filename	language	code	comment	blank	total
src/chains/ethereum/connection.ts	TypeScript	24	0	3	27
src/chains/ethereum/listener.ts	TypeScript	117	4	14	135
src/chains/ethereum/transaction.ts	TypeScript	210	25	36	271
src/chains/ethereum/writer.ts	TypeScript	26	4	7	37
src/chains/substrate/chain.ts	TypeScript	64	7	16	87
src/chains/substrate/connection.ts	TypeScript	34	0	7	41
src/chains/substrate/listener.ts	TypeScript	88	6	17	111
src/chains/substrate/writer.ts	TypeScript	159	9	28	196
src/utils/blockstore.ts	TypeScript	25	0	3	28
src/utils/chain-interface.ts	TypeScript	67	0	6	73
src/utils/core.ts	TypeScript	54	0	12	66
src/utils/msg.ts	TypeScript	40	0	6	46
src/utils/system.ts	TypeScript	34	0	10	44

5.4 Architecture Overview

Ternoa Bridge is built on ChainBridge, a modular Multi-Directional Blockchain Bridge to interact with Ethereum, Substrate, based chains networks. ChainBridge is an extensible cross-chain communication protocol. It currently supports bridging between EVM and Substrate based chains.

A bridge contract (or pallet in Substrate) on each chain forms either side of a bridge. Handler contracts allow for customizable behaviour upon receiving transactions to and from the bridge. For example locking up an asset on one side and minting a new one on the other.



r

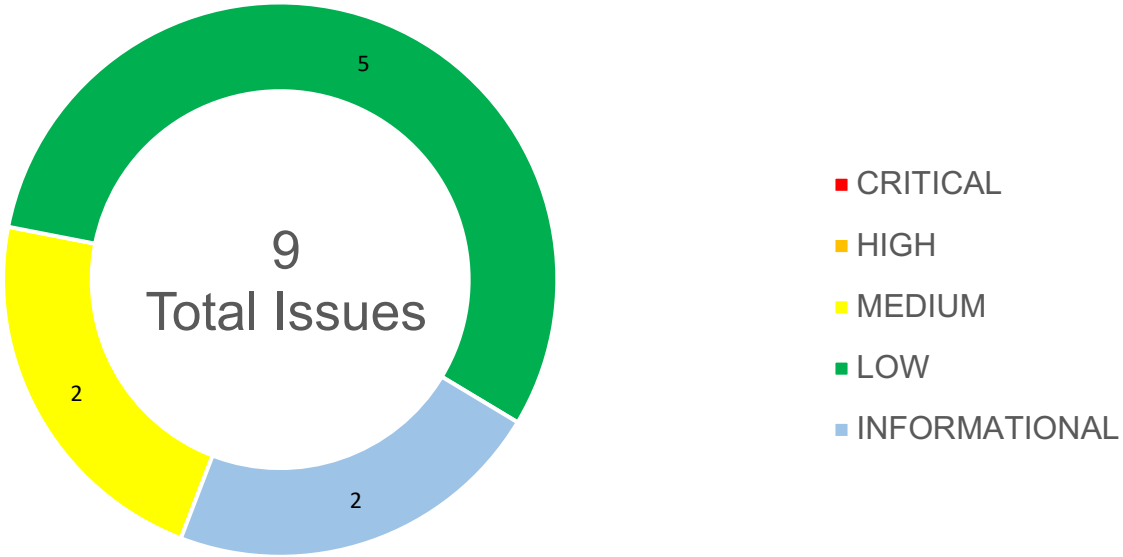
6. Scope of Work

The Ternoa Team provided us with the files that needs to be tested. The scope of the audit is the bridge relayer codebase.

1. Automated Vulnerability Test (OWASP, Acunetix, Sonarsource, Snyk, jsfuzz..)
2. Manual Security Testing (Line by line, Overflow, CVE...)
3. Test environment deployment
4. Evaluating and testing software architecture
   - Development Productivity (Code Conventions Check, packages)
   - Functions & Logic Testing
   - Performance (Connection pooling, caching, Transaction Concurrency)
   - Reliability & Availability
   - Maintainability & Ease of Deployment

The main goal of this audit was to make sure the infrastructure is built according to newest standards and securely developed. The auditors can provide additional feedback on the code upon the client's request.

6.1 Findings Overview



No	Title	Severity	Status
6.2.1	Parameter Pending Is Missing	MEDIUM	FIXED
6.2.2	No Error Handling For pollBlocks()	MEDIUM	FIXED
6.2.3	Incorrect Assertion Of GasPrice And GasLimit	LOW	FIXED
6.2.4	Incorrect Operator Used	LOW	FIXED
6.2.5	Feature Chain ID Problems > 255	LOW	ACKNOWLEDGED
6.2.6	Variables Could Be Declared As Constant	LOW	FIXED
6.2.7	Dead Or Commented Out Code	LOW	FIXED
6.2.8	Parameter Restriction	INFORMATIONAL	ACKNOWLEDGED
6.2.9	Use Different Method Instead Of cmp()	INFORMATIONAL	FIXED

6.2 Manual and Automated Vulnerability Test

CRITICAL ISSUES
During the audit, Chainsulting's experts found **0 Critical issues** in the code of the smart contract.

HIGH ISSUES
During the audit, Chainsulting's experts found **0 High issues** in the code of the smart contract.

MEDIUM ISSUES
During the audit, Chainsulting's experts found **2 Medium issues** in the code of the smart contract.

6.2.1 Parameter Pending Is Missing
Severity: MEDIUM
Status: FIXED
Code: CWE-234
File(s) affected: src/chains/ethereum/transaction.ts
Update: Commit 841f6087bd130fd1619dfb03a1826512f8889b29

Attack / Description	getTransactionCount() should have the second parameter set to "pending". Because the default is "latest" which could result in using a nonce which is in use by a pending transaction. Also keeping an internal count of the current nonce and comparing it with getTransactionCount() can lead to better results in situations where the RPC server might not be fully sync with the blockchain or crashed.
Code	Line 77 - 80 (src/chains/ethereum/transaction.ts) `// Save the next nonce to be used if it's not a replacement`

	if (!replace || !options.nonce) {
	options.nonce = await
	this.conn.provider.getTransactionCount(this.conn.key.address);
	}

Result/Recommendation	We recommend to add a second parameter “pending” to getTransactionCount() and internal count & compare of the current nonce.

6.2.2 No Error Handling For pollBlocks()
Severity: MEDIUM
Status: FIXED
Code: CWE-703
File(s) affected: src/chains/ethereum/listener.ts, src/chains/substrate/listener.ts
Update: Commit 841f6087bd130fd1619dfb03a1826512f8889b29

Attack / Description	If pollBlocks() throws or returns an error, it will just be displayed. There is no error handling to restart pollBlocks() or to exit the process, to have it auto restarted externally. This can lead to bridge errors and outages.

Code	Line 41 - 50 (src/chains/ethereum/listener.ts)
	
	// Start the worker
	(async () => {
	try {
	const res = await this.pollBlocks();
	if (res.isErr) {
	throw res.error;
	}
	} catch (error: any) {
	console.log(`ETHEREUM – Polling blocks failed. Error: ${logError(error)}`);

	}
	})();
	
	Line 41 - 50 (src/chains/substrate/listener.ts)
	
	// Start the worker
	(async () => {
	try {
	const res = await this.pollBlocks();
	if (res.isErr) {
	console.log(`SUBSTRATE - Polling blocks failed. Error: ${logError(res.error)}`);
	}
	} catch (error: any) {
	console.log(`SUBSTRATE - Polling blocks failed. Error: ${logError(error)}`);
	}
	})();

Result/Recommendation	We recommend to add an error handling, to restart pollBlocks() or to exit the process.

LOW ISSUES

During the audit, Chainsulting's experts found **5 Low issues** in the code of the smart contract.

6.2.3 Incorrect Assertion Of GasPrice And GasLimit
Severity: LOW
Status: FIXED

Code: NA
File(s) affected: src/chains/ethereum/transaction.ts,
Update: Commit 841f6087bd130fd1619dfb03a1826512f8889b29

Attack / Description	The types of gasPrice and gasLimit should not be asserted to BigNumber. gasPrice and gasLimit should be checked for being undefined and an error should be thrown if they are. If they are undefined and be used as BigNumber an error would be thrown anyway.

Code	Line 244 (src/chains/ethereum/transaction.ts)
	
	`// Calculate fee`
	`const voteFee = (options.gasPrice as BigNumber).mul(options.gasLimit as BigNumber);`

Result/Recommendation	We recommend to remove the assertion of gasPrice and gasLimit from BigNumber. gasPrice and gasLimit should be checked for being undefined and an error should be thrown if they are.

6.2.4 Incorrect Operator Used

Severity: LOW
Status: FIXED
Code: CWE-480
File(s) affected: src/chains/substrate/writer.ts
Update: Commit 841f6087bd130fd1619dfb03a1826512f8889b29

Attack / Description	An operator defines some function that will be performed on the data. The data on which operators work are called operands. Wrong operators can lead to undefined results.
Code	Line 141 (src/chains/substrate/writer.ts)

	`if (!(new BN(expiry).cmp(finalizedBlockNumber) === 1)) {`
	`return Result.ok([false, "proposal expired"]);`
	`}`

Result/Recommendation	Use the opposite operator (!==) instead.

6.2.5 Feature Chain ID Problems > 255
Severity: LOW
Status: ACKNOWLEDGED
Code: NA
File(s) affected: src/chains/ethereum/transaction.ts,

Attack / Description	If originDomainID (chain ID) is greater than 255 the resulting value will be ambiguous for some depositNonce values.

Code	Line 258 (src/chains/ethereum/transaction.ts)
	
	`private getDestAndNonce(depositNonce: BN, originDomainID: BN): BigNumber {`
	`//Equivalent in solidity : (uint72(depositNonce) << 8) | uint72(originDomainID)`
	`return`
	`BigNumber.from(depositNonce.toString()).shl(8).or(BigNumber.from(originDomainID.toString()));`
	`}`

Result/Recommendation	There's a bunch of other chains using large values for the chain id (https://github.com/ethereum-lists/chains), better to increase depositNonce unit

6.2.6 Variables Could Be Declared As Constant

Severity: LOW
Status: FIXED
Code: NA
File(s) affected: src/chains/substrate/chains.ts, src/chains/ethereum/chain.ts, src/chains/substrate/writer.ts, src/chains/ethereum/transaction.ts, src/chains/substrate/connection.ts, src/utils/core.ts
Update: Commit 841f6087bd130fd1619dfb03a1826512f8889b29

Attack / Description	Variables can be declared using const similar to var or let declarations. The const makes a variable a constant where its value cannot be changed. Const variables have the same scoping rules as let variables. Const variables must be declared and initialized in a single statement.
Code	src/chains/substrate/chains.ts:(24, 43, 47, 55) src/chains/ethereum/chain.ts(25, 44, 48, 55, 65, 75) src/chains/substrate/writer.ts:(30, 38, 44, 51, 67, 120) src/chains/ethereum/transaction.ts:(40, 229) src/chains/substrate/connection.ts:(20, 29) src/utils/core.ts:(54)
Result/Recommendation	We recommend declaring the above variables as constant.

6.2.7 Dead Or Commented Out Code

Severity: LOW
Status: FIXED
Code: CWE-561
File(s) affected: src/chains/ethereum/chain.ts
Update: Commit 841f6087bd130fd1619dfb03a1826512f8889b29

Attack / Description	Programmers should not comment out code as it bloats programs and reduces readability.

Code	Line 77 (src/chains/ethereum/chain.ts) `// Save the next nonce to be used if it's not a replacement`
Result/Recommendation	The comment is false. There is no check. Unused code should be deleted and can be retrieved from source control history if required.

INFORMATIONAL ISSUES

During the audit, Chainsulting's experts found **2 Informational issues** in the code of the smart contract.

6.2.8 Parameter Restriction
Severity: INFORMATIONAL
Status: ACKNOWLEDGED
Code: NA
File(s) affected: src/utils/blockstore.ts

Attack / Description	The parameter "chainName" could be restricted to names of chains that are used to prevent typos on calling the function.

Code	Line 5, 16 (src/utils/blockstore.ts)
	
	`export async function saveBlockNumber(currentBlockNumber: BN, chainName: string): Promise<void> {`
	
	`export async function readBlockNumber(chainName: string): Promise<string | null> {`

Result/Recommendation	We recommend to restrict the parameter to for example chainName: "substrate" | "ethereum"

6.2.9 Use Different Method Instead Of cmp()
Severity: INFORMATIONAL
Status: FIXED
Code: NA
File(s) affected: src/chain/ethereum/listener.ts, src/chains/substrate/writer.ts
Update: Commit 841f6087bd130fd1619dfb03a1826512f8889b29

Attack / Description	Certain methods instead of cmp() will improve the code readability

Code	Line 32, 66 (src/chain/ethereum/listener.ts)
	
	`if (lastBlockNumber.cmp(this.startBlock) === -1) {`
	
	`if (lastBlockNumber.sub(currentBlockNumber).cmp(new BN(BLOCK_CONFIRMATION)) === -1) {`
	
	Line 141, 178 (src/chains/substrate/writer.ts)
	
	`if (!(new BN(expiry).cmp(finalizedBlockNumber) === 1)) {`
	
	`if (balance.cmp(fee) === -1) {`

Result/Recommendation	Using the methods lt(), lte(), gt() and gte() instead of cmp() would improve code readability.

6.3 Excluded Audit Parts

The Ternoa Bridge is based on unchanged code from Chainbridge, which was audited by consensys and was not part of this audit.

Source: https://github.com/ChainSafe/ChainBridge

Documentation: https://chainbridge.chainsafe.io

Audit: https://consensys.net/diligence/audits/private/adash47d-chainbridge/ (May 2020)

7. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the architecture and codebase. The final debriefs took place on the May 19, 2022.

The main goal of this audit was to make sure the bridge relayer is built according to newest standards and securely developed. During the audit, no critical, no high, 2 medium, 5 low and 3 informational issues were found, after the manual and automated security testing.

We recommend the following things to apply to the next dev ops:

i. Check our issues and get them fixed
ii. Make sure newest packages are used, wherever possible
iii. Check for possible centralization risk if you run one relayer
iv. Check open issues from ChainBridge https://github.com/ChainSafe/ChainBridge/issues and fix in your fork if necessary

8. About the Auditor

Chainsulting is a professional software development firm, founded in 20217 and based in Germany. They show ways, opportunities, risks and offer comprehensive blockchain solutions. Some of their services include blockchain development, smart contract audits and consulting.

Chainsulting conducts code audits on market-leading blockchains such as Hyperledger, Tezos, Ethereum, Binance Smart Chain, and Solana to mitigate risk and instil trust and transparency into the vibrant crypto community. They have also reviewed and secure the smart contracts of 1Inch, POA Network, Unicrypt, Amun, Furucombo among numerous other top DeFi projects.

Chainsulting currently secures $100 billion in user funds locked in multiple DeFi protocols. The team behind the leading audit firm relies on their robust technical know-how in the blockchain sector to deliver top-notch smart contract audit solutions, tailored to the clients' evolving business needs.

Check our website for further information: https://chainsulting.de