



**DACXI**

**Token Migration**

**SMART CONTRACT AUDIT**

**13.11.2024**

**Made in Germany by Softstack.io**



## Table of contents

1. Disclaimer.....	4
2. About the Project and Company .....	5
2.1 Project Overview.....	6
3. Vulnerability & Risk Level .....	7
4. Auditing Strategy and Techniques Applied.....	8
4.1 Methodology .....	8
5. Metrics .....	9
5.1 Tested Contract Files .....	9
5.2 CallGraph.....	10
5.3 Inheritance Graph.....	11
5.4 Source Lines & Risk.....	12
5.5 Capabilities .....	13
5.6 Source Unites in Scope .....	14
6. Scope of Work.....	15
6.1 Findings Overview .....	16
6.2 Manual and Automated Vulnerability Test.....	17
6.2.1 Irreversible Whitelist Disabling and Ownership Renouncement.....	17
6.2.2 Lack of Emergency Pause Mechanism .....	19
6.2.3 Unlimited Total Token Supply .....	20
6.2.4 Unlimited Token Migration Amount .....	23
6.2.5 Limited Transparency of Whitelist Status .....	24
6.3 SWC Attacks .....	26



6.4 Unit Tests .....	30
6.5 Verify Claims .....	33
7. Executive Summary.....	34
8. About the Auditor .....	35



## 1. Disclaimer

The audit makes no statements or warranties 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 DACXI PTE. LTD. 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 (07.11.2024)	Layout
0.4 (08.11.2024)	Automated Security Testing Manual Security Testing
0.5 (08.11.2024)	Verify Claims
0.6 (08.11.2024)	Testing SWC Checks
0.9 (08.11.2024)	Summary and Recommendation
1.0 (08.11.2024)	Final document
1.1 (13.11.2024)	Re-check <a href="https://github.com/dacxi/dacxi-coin-contract/pull/7/commits">https://github.com/dacxi/dacxi-coin-contract/pull/7/commits</a>

## 2. About the Project and Company

### Company address:

GCF Technologies Pty Ltd.  
New South Wales (NSW) 2060  
Australia

**dacxi**chain®

**Website:** <https://dacxichain.com>

**LinkedIn:** <https://www.linkedin.com/company/dacxiglobal>

**Twitter (X):** <https://twitter.com/dacxicoins>

**Reddit:** <https://www.reddit.com/r/DACXI>

**Telegram:** <https://t.me/DacxiCoinCommunity>

**Medium:** <https://dacxi.medium.com>

**Youtube:** <https://www.youtube.com/c/Dacxi>

**Facebook:** <https://www.facebook.com/DacxiChain>



## 2.1 Project Overview

The Dacxi Chain is a blockchain-based platform that seeks to redefine global equity crowdfunding by making international investments more accessible and transparent. Its mission is to connect investors with entrepreneurs on a global scale, enabling individuals from diverse backgrounds to participate in funding projects that were previously out of reach due to geographic and regulatory barriers. Central to this ecosystem is the Dacxi Token (DACXI), a utility token specifically designed to drive functionality within the Dacxi Chain.

The DACXI token facilitates a range of essential operations, including covering blockchain transaction fees, enabling seamless cross-border investment transfers, and acting as the primary currency for transactions within the platform. Beyond its core transactional role, the DACXI token is a critical element of the platform's broader vision for democratizing investment opportunities. By leveraging the DACXI token, the platform aims to provide a secure, efficient, and cost-effective solution for managing investments across different jurisdictions, all while maintaining compliance with global regulatory standards. This unique approach allows users to overcome traditional challenges in equity crowdfunding, such as high transaction costs and limited accessibility, ultimately fostering a more inclusive global investment ecosystem.

By integrating blockchain technology and focusing on regulatory compliance, the Dacxi Chain positions itself as a leader in decentralized finance, paving the way for a future where investment opportunities are universally accessible and empowering a new generation of global investors.

### 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 contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract 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 contract 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 contract 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 pentesters and smart contract 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 softstack to make sure we understand the size, scope, and functionality of the smart contract.
  - 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 softstack 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 smart contracts 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 smart contracts.



## 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 Contract Files

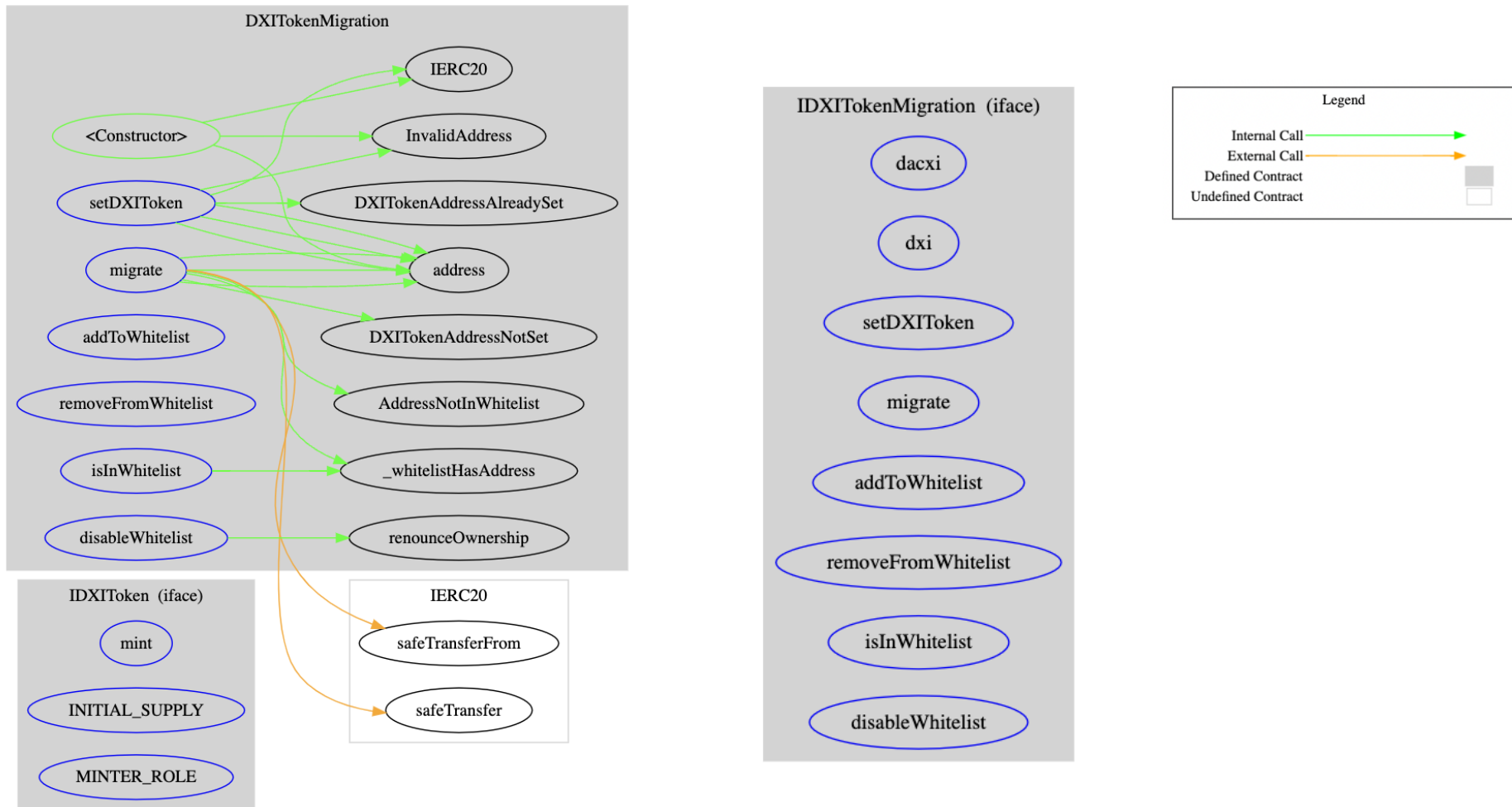
The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability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

File	Fingerprint (MD5)
./src/DXIToken.sol	d5be80e19631db7da91878b1a1a780fd
./src/DXITokenMigration.sol	4d6c072a849ca1c022798f6b5271c3bf
./src/interfaces/IDXIToken.sol	3d92807e0944a5cdc0613ec3dbfdbe47
./src/interfaces/IDXITokenMigration.sol	a7d2b263ea0a412b28d4e5e0a50112ab

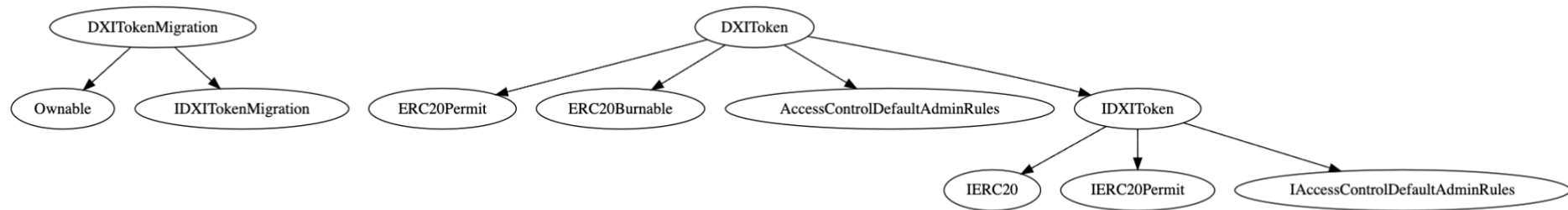
Updated (<https://github.com/dacxi/dacxi-coin-contract/commit/8bf86fb92c8dc303f5b1b3c13038e9f75d6af4e7>)

File	Fingerprint (MD5)
./src/DXIToken.sol	d5be80e19631db7da91878b1a1a780fd
./src/DXITokenMigration.sol	1c38b21e745c001798709d0821462bff
./src/interfaces/IDXIToken.sol	6e8d4067c5fb8f44863f23eb01d374f8
./src/interfaces/IDXITokenMigration.sol	ae6be06d51852db33d8788a535ba82c6

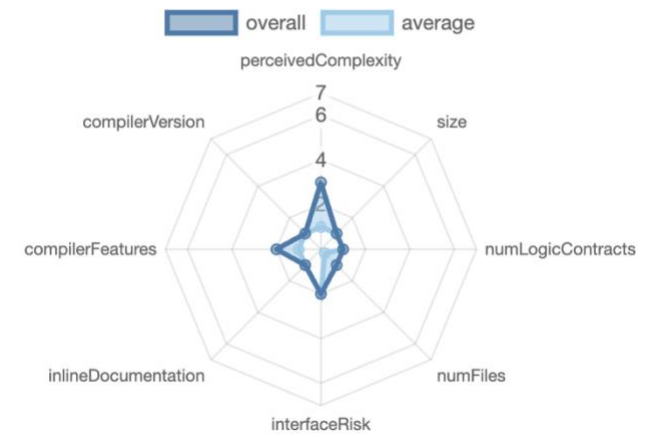
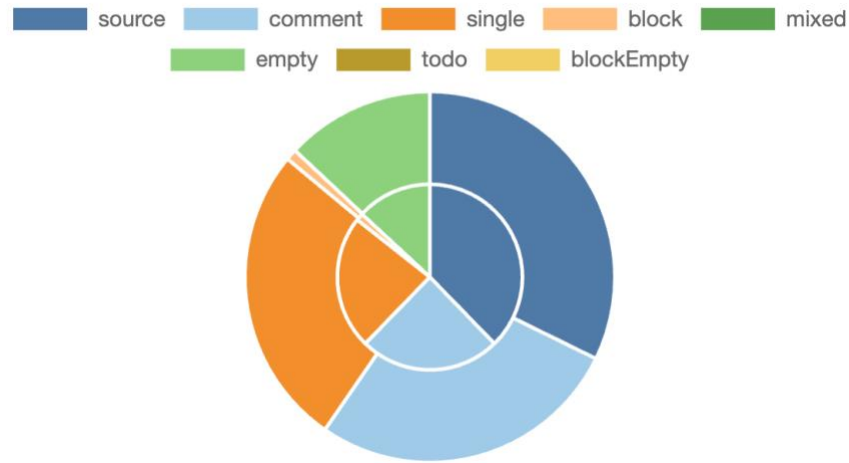
## 5.2 CallGraph













## 5.3 Inheritance Graph





## 5.4 Source Lines & Risk



## 5.5 Capabilities


Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts	
^0.8.27					
 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRrecover	 New/Create/Create2
			yes		

Exposed Functions  
This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

 Public	 Payable
19	0

External	Internal	Private	Pure	View
17	14	1	0	8

### StateVariables

Total	 Public
8	4



## 5.6 Source Unites in Scope

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complexity Score
src/DXITokenMigration.sol	1		81	81	45	18	53
src/DXIToken.sol	1		49	49	27	15	26
src/interfaces/IDXITokenMigration.sol		1	65	29	10	37	17
src/interfaces/IDXIToken.sol		1	28	19	7	14	13
<b>Totals</b>	<b>2</b>	<b>2</b>	<b>223</b>	<b>178</b>	<b>89</b>	<b>84</b>	<b>109</b>

- **Lines:** total lines of the source unit
- **nLines:** normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC:** normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines:** lines containing single or block comments
- **Complexity Score:** a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

## 6. Scope of Work

The Dacxi team has provided the necessary files for auditing, with the primary focus on the Token Migration contracts. The scope of this audit includes verifying key security, functionality, and efficiency aspects of the contracts, following established standards and best practices. The audit will address the following critical areas:

1. **Compliance with Smart Contract Best Practices:** The audit will examine the Token Migration contracts to ensure they adhere to industry-standard smart contract practices. The goal is to confirm that the contracts are resilient against these common security threats, ensuring robustness and reliability throughout the migration process.
2. **Access Control and Permissions:** The contracts will be reviewed for strict access control mechanisms to guarantee that only authorized entities can access or modify sensitive migration functions. This audit phase aims to prevent unauthorized actions that could compromise the token migration, ensuring that only designated roles or users can execute migration-related operations.
3. **Gas Efficiency:** The contracts will be evaluated for gas optimization to minimize computational costs and unnecessary storage operations, making the migration process more efficient. Gas efficiency is essential to reducing transaction fees, allowing for a smoother and more cost-effective migration experience for users.
4. **Security of Migration Process:** The audit will thoroughly examine the migration mechanics to ensure the process is secure and follows a 1-to-1 token conversion ratio. Special attention will be given to preventing unauthorized migrations, abuses of whitelisting functions, and any manipulation of the migration controls.

The primary goal of this audit is to verify the security and efficiency of the Token Migration contracts, providing assurance that they are secure, optimized, and reliable. Upon request from the Dacxi team, the audit team can offer additional insights or feedback on specific areas within the migration contracts, addressing any unique concerns or requirements. This comprehensive audit will help ensure that the Token Migration contracts are prepared for a secure, efficient, and trustworthy deployment within the Dacxi ecosystem.



6.1 Findings Overview



No	Title	Severity	Status
6.2.1	Irreversible Whitelist Disabling and Ownership Renouncement	LOW	FIXED
6.2.2	Lack of Emergency Pause Mechanism	LOW	ACKNOWLEDGED
6.2.3	Unlimited Total Token Supply	INFORMATIONAL	ACKNOWLEDGED
6.2.4	Unlimited Token Migration Amount	INFORMATIONAL	ACKNOWLEDGED
6.2.5	Limited Transparency of Whitelist Status	INFORMATIONAL	FIXED





## 6.2 Manual and Automated Vulnerability Test

### CRITICAL ISSUES

During the audit, softstack's experts found **no Critical issues** in the code of the smart contract.

### HIGH ISSUES

During the audit, softstack's experts found **no High issues** in the code of the smart contract.

### MEDIUM ISSUES

During the audit, softstack's experts found **no Medium issues** in the code of the smart contract.

### LOW ISSUES

During the audit, softstack's experts found **2 Low issues** in the code of the smart contract

#### 6.2.1 Irreversible Whitelist Disabling and Ownership Renouncement

Severity: LOW

Status: FIXED

File(s) affected: DXITokenMigration.sol

Update: <https://github.com/dacxi/dacxi-coin-contract/pull/7/commits>

Attack / Description	The DXITokenMigration contract includes a function disableWhitelist() that permanently disables the whitelist and simultaneously renounces contract ownership. This design creates a critical, irreversible state change that could lead to significant issues if executed prematurely or accidentally. Once called, there's no way to re-enable the whitelist or regain ownership, potentially leaving the contract in an unmanageable state.
----------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



Code	<p>Line 67 – 72 (DXITokenMigration.sol):</p> <pre> /// @inheritdoc IDXITokenMigration function disableWhitelist() external onlyOwner {     isWhitelistEnabled = false;      renounceOwnership(); } </pre>
Result/Recommendation	<ol style="list-style-type: none"> <li>1. Separate the whitelist disabling and ownership renouncement into two distinct functions to ensure clear and independent management of these actions.</li> <li>2. Implement a time-lock mechanism for the whitelist disabling to allow a grace period before the change takes effect, providing an additional layer of security.</li> <li>3. Add a confirmation step for ownership renouncement, such as requiring the owner to call the function twice with a time delay between calls to prevent accidental renouncement.</li> <li>4. Consider implementing a pausable pattern to allow temporary disabling of the whitelist without making permanent changes, enabling flexibility in managing access controls.</li> <li>5. Add extensive logging and event emissions for these critical state changes to ensure transparency and provide an audit trail for all significant actions.</li> </ol> <pre> uint256 public constant WHITELIST_DISABLE_DELAY = 7 days; uint256 public whitelistDisableTime;  function initiateWhitelistDisable() external onlyOwner {     whitelistDisableTime = block.timestamp + WHITELIST_DISABLE_DELAY;     emit WhitelistDisableInitiated(whitelistDisableTime); }  function finalizeWhitelistDisable() external onlyOwner {     require(block.timestamp &gt;= whitelistDisableTime, "Disable delay not elapsed");     require(whitelistDisableTime != 0, "Disable not initiated"); } </pre>

	<pre> isWhitelistEnabled = false; emit WhitelistDisabled(); } function cancelWhitelistDisable() external onlyOwner {     whitelistDisableTime = 0;     emit WhitelistDisableCancelled(); } </pre>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 6.2.2 Lack of Emergency Pause Mechanism

Severity: LOW

Status: ACKNOWLEDGED

File(s) affected: DXIToken.sol

<b>Attack / Description</b>	The DXIToken contract does not implement a pause mechanism, which is a crucial safety feature for many token contracts. Without a pause functionality, the contract lacks the ability to temporarily halt critical operations such as minting, transfers, or burns in case of emergencies, such as discovered vulnerabilities or ongoing attacks. This absence of a pause feature could potentially lead to continued exploitation in case of a security breach, making it difficult to mitigate risks quickly.
<b>Code</b>	NA
<b>Result/Recommendation</b>	<p>Implement a pause mechanism in the contract. This can be achieved by:</p> <ol style="list-style-type: none"> <li>1. Integrate OpenZeppelin's Pausable contract by importing it into your contract file: import "@openzeppelin/contracts/security/Pausable.sol";</li> <li>2. Inherit from Pausable in the contract declaration to enable pausing functionality: contract YourContractName is Pausable {</li> </ol>

	<pre>// Contract code here }</pre> <p>3. Add pause and unpause functions with appropriate access control to manage the paused state of the contract:</p> <pre>function pause() public onlyRole(DEFAULT_ADMIN_ROLE) {     _pause(); }  function unpause() public onlyRole(DEFAULT_ADMIN_ROLE) {     _unpause(); }</pre> <p>4. Modify critical functions to include the whenNotPaused modifier, ensuring they can only be executed when the contract is not paused:</p> <pre>function criticalFunction() public whenNotPaused {     // Critical function logic here }</pre> <p>5. Consider adding the whenNotPaused modifier to functions like transfer as well to prevent token transfers while the contract is paused:</p> <pre>function transfer(address recipient, uint256 amount) public whenNotPaused returns (bool) {     // Transfer logic here }</pre>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## INFORMATIONAL ISSUES

During the audit, softstack's experts found **3 Informational issues** in the code of the smart contract.

### 6.2.3 Unlimited Total Token Supply

Severity: INFORMATIONAL



Status: ACKNOWLEDGED

File(s) affected: DXIToken.sol

Update: To provide further clarity on the DXIToken emission mechanism and address the issue raised:

1. **Annual Emission:** The DXIToken will have an annual emission model. However, specific details, such as the start date and the exact process for minting new tokens, are still under consideration and not yet defined.
2. **Emission Management:** Emission will be controlled by an external smart contract, which will have the exclusive authority to mint new tokens. The DXIToken contract specifies that the emission rate cannot exceed a cap of 45 tokens per second. This is a hard limit, but it does not imply that emission will necessarily occur at this rate.
3. **Role-Based Access Control:**
  - **MINTER\_ROLE:** The designated emission smart contract will hold the MINTER\_ROLE, granting it the authority to mint new tokens.
  - **DEFAULT\_ADMIN\_ROLE:** This role, which can modify and assign other roles (like the MINTER\_ROLE), will be assigned to a DAO address—a multi-signature wallet controlled by the DAO. This setup ensures that the community has governance over key aspects of token issuance.

*Note:* Although only one smart contract will manage the emission at any given time, we chose a role-based mechanism for flexibility and ease of future adjustments, as detailed below.

4. **Distribution of Emittted Tokens:** Emission and distribution will be managed on Ethereum Layer 1. The emission smart contract will define how new tokens are allocated, such as directing portions to staking vaults, liquidity incentives, etc. The DAO will retain the authority to modify this distribution as necessary.
5. **Future Adjustments to Emission Rules:** The architecture allows for flexibility. The DAO has the ability to transfer the MINTER\_ROLE to another address (smart contract) in the future if it decides that the emission rate needs adjustment or if token emission should cease. Any new emission contract will still be bound by the universal cap of 45 tokens per second.
6. **Option to Make Emission Rules Immutable:** If desired, the DAO can make the emission rules immutable by renouncing the ADMIN\_ROLE from the DXIToken contract. In doing so, the current emission smart contract would become permanent, with no option for future changes. This possibility was also considered during the architectural design.

These mechanisms provide flexibility for future adjustments while ensuring adherence to the emission cap, aligning with both governance and security considerations.

<b>Attack / Description</b>	<p>The DXIToken contract implements a minting mechanism with a per-second cap (MAX_MINT_CAP) but does not include an absolute cap on the total token supply. This design allows for continuous token minting over time without an upper limit, potentially leading to unlimited token inflation. While the contract controls the minting rate, the absence of a total supply cap could result in an ever-increasing token supply, which may impact the token's economics and value over time.</p>
<b>Code</b>	<p>Line 34 - 37 (DXIToken.sol):</p> <pre> /// @inheritdoc IDXIToken function mint(address to, uint256 amount) public onlyRole(MINTER_ROLE) {     _mint(to, amount); } </pre>
<b>Result/Recommendation</b>	<p>Implement a maximum total supply cap to limit the overall number of tokens that can be minted. This can be done by:</p> <ol style="list-style-type: none"> <li>1. Adding a new state variable for the maximum total supply:  uint256 public constant MAX_TOTAL_SUPPLY = X; // Set appropriate value</li> <li>1. Modifying the mint function to check against this cap:  <pre> function mint(address to, uint256 amount) public onlyRole(MINTER_ROLE) {     // Existing checks...     require(totalSupply() + amount &lt;= MAX_TOTAL_SUPPLY, "Max total supply exceeded");     // Rest of the minting logic... } </pre> </li> <li>2. Consider adding a public function to check the remaining mintable supply:  <pre> function remainingSupply() public view returns (uint256) {     return MAX_TOTAL_SUPPLY - totalSupply(); } </pre> </li> </ol>

6.2.4 Unlimited Token Migration Amount

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

File(s) affected: DXITokenMigration.sol

Attack / Description	<p>The DXITokenMigration contract currently allows users to migrate an unlimited amount of tokens in a single transaction. The migrate function does not impose any upper limit on the amount parameter, which may permit unexpectedly large migrations. This could lead to several potential issues:</p> <ol style="list-style-type: none"><li>1. Gas Limit Exceedance: Extremely large migrations might exceed the block gas limit, causing transactions to fail. This could disrupt user experience and prevent successful migrations.</li><li>2. Token Supply Imbalance: A single, large migration could significantly impact the token supply distribution between the source and destination chains. Such imbalances may lead to liquidity issues or affect token availability on either chain.</li><li>3. Potential for Economic Attacks: Malicious actors could exploit the lack of migration limits to manipulate token markets or conduct large-scale arbitrage. This could have serious consequences for the token’s economic stability and market dynamics.</li></ol>
Code	<p>Line 42 - 50 (DXITokenMigration.sol):</p> <pre>function migrate(uint256 amount) external {     if (address(dxi) == address(0)) revert DXITokenAddressNotSet();     if (!_whitelistHasAddress(msg.sender)) revert AddressNotInWhitelist();      emit Migrated(msg.sender, amount);      dacxi.safeTransferFrom(msg.sender, address(this), amount);     dxi.safeTransfer(msg.sender, amount); }</pre>



	}
<b>Result/Recommendation</b>	<p>To implement a maximum migration limit per transaction in the DXITokenMigration contract, follow these steps:</p> <ol style="list-style-type: none"> <li>1. Add a state variable to store the maximum migration amount allowed per transaction:  uint256 public maxMigrationAmount;</li> <li>2. Implement a function to set and update this migration limit. This function should have restricted access, allowing only the contract owner to modify the limit:  function setMaxMigrationAmount(uint256 _maxAmount) external onlyOwner  { maxMigrationAmount = _maxAmount; }</li> <li>3. Modify the migrate function to include a check against the maximum migration limit. This ensures that the migration amount does not exceed the allowed limit:  function migrate(uint256 amount) external {  // ... existing checks ...  require(amount &lt;= maxMigrationAmount, "Exceeds maximum migration amount");  // ... rest of the function logic ...  }</li> </ol>

#### 6.2.5 Limited Transparency of Whitelist Status

Severity: INFORMATIONAL

Status: FIXED

File(s) affected: DXITokenMigration.sol

Update: <https://github.com/dacxi/dacxi-coin-contract/pull/7/commits>



<b>Attack / Description</b>	<p>In the DXITokenMigration contract, the <code>isInWhitelist</code> function is restricted to be called only by the contract owner. This design choice significantly limits transparency and accessibility for users who want to check their own whitelist status.</p> <p>This implementation prevents users from directly querying their whitelist status, forcing them to rely on the contract owner or off-chain methods to obtain this information. Such a restriction can lead to a lack of transparency in the migration process and potentially reduce user trust in the system.</p>
<b>Code</b>	<p>Line 63 - 65 (DXITokenMigration.sol):</p> <pre>function isInWhitelist(address account) external view onlyOwner returns (bool) {     return _whitelistHasAddress(account); }</pre>
<b>Result/Recommendation</b>	<p>We recommend the following changes:</p> <ol style="list-style-type: none"> <li>1. Remove the <code>onlyOwner</code> modifier from the <code>isInWhitelist</code> function to allow public access, enabling anyone to check if a specific address is in the whitelist: <pre>function isInWhitelist(address account) external view returns (bool) {     return _whitelistHasAddress(account); }</pre> </li> <li>2. Implement a function to allow users to check only their own whitelist status, addressing any concerns about exposing the entire whitelist: <pre>function checkMyWhitelistStatus() external view returns (bool) {     return _whitelistHasAddress(msg.sender); }</pre> </li> <li>3. Add an event emission system to notify users when they are added to or removed from the whitelist, providing better transparency and traceability: <pre>event WhitelistStatusChanged(address indexed account, bool isWhitelisted);</pre> </li> </ol>

	<p>4. Define functions to add and remove addresses from the whitelist, including event emissions to signal these changes:</p> <pre> function addToWhitelist(address account) external onlyOwner {     whitelist[account] = true;     emit WhitelistStatusChanged(account, true); } function removeFromWhitelist(address account) external onlyOwner {     whitelist[account] = false;     emit WhitelistStatusChanged(account, false); } </pre>
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 6.3 SWC Attacks

ID	Title	Relationships	Test Result
<a href="#">SWC-131</a>	Presence of unused variables	<a href="#">CWE-1164: Irrelevant Code</a>	✓
<a href="#">SWC-130</a>	Right-To-Left-Override control character (U+202E)	<a href="#">CWE-451: User Interface (UI) Misrepresentation of Critical Information</a>	✓
<a href="#">SWC-129</a>	Typographical Error	<a href="#">CWE-480: Use of Incorrect Operator</a>	✓
<a href="#">SWC-128</a>	DoS With Block Gas Limit	<a href="#">CWE-400: Uncontrolled Resource Consumption</a>	✓

ID	Title	Relationships	Test Result
<a href="#">SWC-127</a>	Arbitrary Jump with Function Type Variable	<a href="#">CWE-695: Use of Low-Level Functionality</a>	✓
<a href="#">SWC-125</a>	Incorrect Inheritance Order	<a href="#">CWE-696: Incorrect Behavior Order</a>	✓
<a href="#">SWC-124</a>	Write to Arbitrary Storage Location	<a href="#">CWE-123: Write-what-where Condition</a>	✓
<a href="#">SWC-123</a>	Requirement Violation	<a href="#">CWE-573: Improper Following of Specification by Caller</a>	✓
<a href="#">SWC-122</a>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	✓
<a href="#">SWC-121</a>	Missing Protection against Signature Replay Attacks	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	✓
<a href="#">SWC-120</a>	Weak Sources of Randomness from Chain Attributes	<a href="#">CWE-330: Use of Insufficiently Random Values</a>	✓
<a href="#">SWC-119</a>	Shadowing State Variables	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	✓
<a href="#">SWC-118</a>	Incorrect Constructor Name	<a href="#">CWE-665: Improper Initialization</a>	✓
<a href="#">SWC-117</a>	Signature Malleability	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	✓

ID	Title	Relationships	Test Result
<a href="#">SWC-116</a>	Timestamp Dependence	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	✓
<a href="#">SWC-115</a>	Authorization through tx.origin	<a href="#">CWE-477: Use of Obsolete Function</a>	✓
<a href="#">SWC-114</a>	Transaction Order Dependence	<a href="#">CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</a>	✓
<a href="#">SWC-113</a>	DoS with Failed Call	<a href="#">CWE-703: Improper Check or Handling of Exceptional Conditions</a>	✓
<a href="#">SWC-112</a>	Delegatecall to Untrusted Callee	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	✓
<a href="#">SWC-111</a>	Use of Deprecated Solidity Functions	<a href="#">CWE-477: Use of Obsolete Function</a>	✓
<a href="#">SWC-110</a>	Assert Violation	<a href="#">CWE-670: Always-Incorrect Control Flow Implementation</a>	✓
<a href="#">SWC-109</a>	Uninitialized Storage Pointer	<a href="#">CWE-824: Access of Uninitialized Pointer</a>	✓
<a href="#">SWC-108</a>	State Variable Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	✓
<a href="#">SWC-107</a>	Reentrancy	<a href="#">CWE-841: Improper Enforcement of Behavioral Workflow</a>	✓

ID	Title	Relationships	Test Result
<a href="#">SWC-106</a>	Unprotected SELFDESTRUCT Instruction	<a href="#">CWE-284: Improper Access Control</a>	✓
<a href="#">SWC-105</a>	Unprotected Ether Withdrawal	<a href="#">CWE-284: Improper Access Control</a>	✓
<a href="#">SWC-104</a>	Unchecked Call Return Value	<a href="#">CWE-252: Unchecked Return Value</a>	✓
<a href="#">SWC-103</a>	Floating Pragma	<a href="#">CWE-664: Improper Control of a Resource Through its Lifetime</a>	✓
<a href="#">SWC-102</a>	Outdated Compiler Version	<a href="#">CWE-937: Using Components with Known Vulnerabilities</a>	✓
<a href="#">SWC-101</a>	Integer Overflow and Underflow	<a href="#">CWE-682: Incorrect Calculation</a>	✓
<a href="#">SWC-100</a>	Function Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	✓

## 6.4 Unit Tests

Ran 21 tests for test/DXIToken.t.sol:DXITokenTest

[PASS] test\_Burn(address,uint256) (runs: 256,  $\mu$ : 37865,  $\sim$ : 37967)

[PASS] test\_BurnFromAccountWithoutBalance(address) (runs: 256,  $\mu$ : 26564,  $\sim$ : 26564)

[PASS] test\_CanMintOncePerSecond(address,uint256,uint72) (runs: 256,  $\mu$ : 174178,  $\sim$ : 174178)

[PASS] test\_CanUpdateMintCapMultipleTimes(address,uint72,uint72,uint72,uint8) (runs: 256,  $\mu$ : 116634,  $\sim$ : 117334)

[PASS] test\_CannotSetAnotherAdmin(address) (runs: 256,  $\mu$ : 10731,  $\sim$ : 10731)

[PASS] test\_InitialSupply() (gas: 12751)

[PASS] test\_Mint(address,uint256,uint72,uint8) (runs: 256,  $\mu$ : 147647,  $\sim$ : 147647)

[PASS] test\_MintRevertIfAmountExceedsCap(address,uint72,uint256,uint8) (runs: 256,  $\mu$ : 108362,  $\sim$ : 109218)

[PASS] test\_MintRevertIfNoCapsSet(address,uint256,uint8) (runs: 256,  $\mu$ : 52395,  $\sim$ : 52395)

[PASS] test\_MintRevertWithoutPermission(address,uint256) (runs: 256,  $\mu$ : 18179,  $\sim$ : 18179)

[PASS] test\_MinterRoleCanBeGranted(address) (runs: 256,  $\mu$ : 42543,  $\sim$ : 42543)

[PASS] test\_MinterRoleCanBeRenouncedAfterAdminRenouce(address,uint48) (runs: 256,  $\mu$ : 58724,  $\sim$ : 58722)

[PASS] test\_MinterRoleCanBeRevoked(address) (runs: 256,  $\mu$ : 36201,  $\sim$ : 36186)

[PASS] test\_MinterRoleCannotBeGrantedAfterAdminRenouce(address,uint48) (runs: 256,  $\mu$ : 36042,  $\sim$ : 36042)

[PASS] test\_MinterRoleCannotBeRevokedAfterAdminRenouce(address,uint48) (runs: 256,  $\mu$ : 61226,  $\sim$ : 61226)

[PASS] test\_Name() (gas: 13461)

[PASS] test\_Nonce() (gas: 11735)

[PASS] test\_RenounceDefaultAdminRoleRequiresTwoSteps(uint48,uint48) (runs: 256,  $\mu$ : 36692,  $\sim$ : 36692)

[PASS] test\_Symbol() (gas: 13504)

[PASS] test\_UpdateMintCap(address,uint72) (runs: 256,  $\mu$ : 100208,  $\sim$ : 101064)

[PASS] test\_UpdateMintCapRevertsWhenExceedsMaxMintCap(address,uint72) (runs: 256,  $\mu$ : 76023,  $\sim$ : 76023)

Suite result: ok. 21 passed; 0 failed; 0 skipped; finished in 4.43s (4.20s CPU time)

Ran 24 tests for test/DXITokenMigration.t.sol:DXITokenMigrationTest

[PASS] test\_AnyoneCanMigrateMultipleTimesWhenWhitelistIsDisabled(address,address,uint256) (runs: 256,  $\mu$ : 958,  $\sim$ : 958)

[PASS] test\_AnyoneCanMigrateWhenWhitelistIsDisabled(address,address,uint256) (runs: 256,  $\mu$ : 209550,  $\sim$ : 212785)

[PASS] test\_CanAddAccountToWhitelistMultipleTimes(address,address) (runs: 256,  $\mu$ : 54678,  $\sim$ : 54678)

[PASS] test\_CanAddMultipleAccountToWhitelist(address,address,address) (runs: 256,  $\mu$ : 75568,  $\sim$ : 75568)

[PASS] test\_ForbiddingCreatingContractWithoutValidDacxiAddress() (gas: 83008)

[PASS] test\_ForbiddingSetDXITokenTo0Address() (gas: 11313)

[PASS] test\_IsInWhitelist(address,address) (runs: 256,  $\mu$ : 46173,  $\sim$ : 46173)

[PASS] test\_Migrate(uint256) (runs: 256,  $\mu$ : 142202,  $\sim$ : 143089)

[PASS] test\_MigrateDacxiTotalSupply() (gas: 141536)

[PASS] test\_MigrateDacxiTotalSupplyWhenWhitelistIsDisabled(address,address,address,uint256) (runs: 256,  $\mu$ : 306464,  $\sim$ : 310050)

[PASS] test\_MigrateMultipleTimes(uint256,uint8) (runs: 256,  $\mu$ : 1890010,  $\sim$ : 1606513)

[PASS] test\_NothingHappensWhenRemovingFromWhitelistANonWhitelistedaddress(address) (runs: 256,  $\mu$ : 23427,  $\sim$ : 23427)

[PASS] test\_OnlyOwnerCanAddToWhitelist(address,address) (runs: 256,  $\mu$ : 22318,  $\sim$ : 22318)

[PASS] test\_OnlyOwnerCanCheckWhitelist(address,address) (runs: 256,  $\mu$ : 44335,  $\sim$ : 44335)



[PASS] test\_OnlyOwnerCanRemoveFromWhitelist(address,address) (runs: 256,  $\mu$ : 37814,  $\sim$ : 37814)

[PASS] test\_OnlyOwnerCanSetDXIToken(address,address) (runs: 256,  $\mu$ : 13588,  $\sim$ : 13588)

[PASS] test\_RemoveFromWhitelist(address,address) (runs: 256,  $\mu$ : 39380,  $\sim$ : 39348)

[PASS] test\_RenouncesTheOwnershipWhenDisableWhitelist(address) (runs: 256,  $\mu$ : 27437,  $\sim$ : 27437)

[PASS] test\_RevertIfSetDXITokenMoreThanOnce(address) (runs: 256,  $\mu$ : 24671,  $\sim$ : 24671)

[PASS] test\_RevertMigrateIfDXITokenIsNotSet(uint256) (runs: 256,  $\mu$ : 32652,  $\sim$ : 32652)

[PASS] test\_RevertMigrateIfNotInWhitelist(uint256) (runs: 256,  $\mu$ : 99052,  $\sim$ : 99885)

[PASS] test\_RevertMigrateWhenAmountMoreThanBalance(uint256,uint256) (runs: 256,  $\mu$ : 128445,  $\sim$ : 128445)

[PASS] test\_SetDXIToken(address) (runs: 256,  $\mu$ : 21728,  $\sim$ : 21728)

[PASS] test\_WhitelistStartsEmpty(address,address,address) (runs: 256,  $\mu$ : 26215,  $\sim$ : 26215)


Suite result: ok. 24 passed; 0 failed; 0 skipped; finished in 9.05s (10.30s CPU time)

File	% Lines	% Statements	% Branches	% Funcs
src/DXIToken.sol	100.00% (11/11)	100.00% (16/16)	100.00% (2/2)	100.00% (4/4)
src/DXITokenMigration.sol	100.00% (21/21)	100.00% (27/27)	100.00% (5/5)	100.00% (8/8)
test/fixtures/DACXI.sol	100.00% (1/1)	100.00% (1/1)	100.00% (0/0)	100.00% (1/1)
Total	100.00% (33/33)	100.00% (44/44)	100.00% (7/7)	100.00% (13/13)




## 6.5 Verify Claims


**6.5.1 Compliance with Smart Contract Best Practices:** The audit will examine the Token Migration contracts to ensure they adhere to industry-standard smart contract practices. The goal is to confirm that the contracts are resilient against these common security threats, ensuring robustness and reliability throughout the migration process.

**Status:** tested and verified 


**6.5.2 Access Control and Permissions:** The contracts will be reviewed for strict access control mechanisms to guarantee that only authorized entities can access or modify sensitive migration functions. This audit phase aims to prevent unauthorized actions that could compromise the token migration, ensuring that only designated roles or users can execute migration-related operations.

**Status:** tested and verified 

**6.5.3 Gas Efficiency:** The contracts will be evaluated for gas optimization to minimize computational costs and unnecessary storage operations, making the migration process more efficient. Gas efficiency is essential to reducing transaction fees, allowing for a smoother and more cost-effective migration experience for users.

**Status:** tested and verified 

**6.5.4 Security of Migration Process:** The audit will thoroughly examine the migration mechanics to ensure the process is secure and follows a 1-to-1 token conversion ratio. Special attention will be given to preventing unauthorized migrations, abuses of whitelisting functions, and any manipulation of the migration controls.

**Status:** tested and verified 

## 7. Executive Summary

Two independent experts from Softstack conducted an unbiased and isolated audit of the smart contract codebase provided by the Dacxi team. The primary objective of this audit was to verify the security, functionality, and adherence to best practices within the smart contracts. The audit process included a comprehensive manual code review, along with automated security testing to detect any potential vulnerabilities.

The audit identified a total of five issues, classified as follows:

- **No critical issues** were found.
- **No high severity issues** were found.
- **No medium severity issue** was identified.
- **Two low severity issues** were discovered.
- **Three informational issues** were noted.

This audit report provides detailed descriptions for each identified issue, including their severity levels and recommendations for mitigation. Code snippets are included where applicable to illustrate the issues and offer possible fixes. We recommend that the Dacxi team carefully review these suggestions to improve the security and robustness of their smart contracts.

Update (13.11.2024): We are pleased to report that all identified issues have been thoroughly addressed. Clarifications have been provided regarding the minting process to ensure transparency and understanding of the token emission mechanism. All necessary fixes have been implemented to enhance the security and functionality of the system.

## 8. About the Auditor

Established in 2017 under the name Chainsulting, and rebranded as softstack GmbH in 2023, softstack has been a trusted name in Web3 Security space. Within the rapidly growing Web3 industry, softstack provides a comprehensive range of offerings that include software development, cybersecurity, and consulting services. Softstack's competency extends across the security landscape of prominent blockchains like Solana, Tezos, TON, Ethereum and Polygon. The company is widely recognized for conducting thorough code audits aimed at mitigating risk and promoting transparency.

The firm's proficiency lies particularly in assessing and fortifying smart contracts of leading DeFi projects, a testament to their commitment to maintaining the integrity of these innovative financial platforms. To date, softstack plays a crucial role in safeguarding over \$100 billion worth of user funds in various DeFi protocols.

Underpinned by a team of industry veterans possessing robust technical knowledge in the Web3 domain, softstack offers industry-leading smart contract audit services. Committed to evolving with their clients' ever-changing business needs, softstack's approach is as dynamic and innovative as the industry it serves.

Check our website for further information: <https://softstack.io>

### How We Work



**1** -----

#### PREPARATION

Supply our team with audit ready code and additional materials



**2** -----

#### COMMUNICATION

We setup a real-time communication tool of your choice or communicate via e-mails.



**3** -----

#### AUDIT

We conduct the audit, suggesting fixes to all vulnerabilities and help you to improve.



**4** -----

#### FIXES

Your development team applies fixes while consulting with our auditors on their safety.



**5** -----

#### REPORT

We check the applied fixes and deliver a full report on all steps done.

