NEXT Token

SMART CONTRACT AUDIT

01.03.2021

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 NEXT Exchange B.V. 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 (26.02.2021)	Layout
0.5 (27.02.2021)	Automated Security Testing Manual Security Testing
0.8 (27.02.2021)	Testing SWC Checks
0.9 (28.02.2021)	Verify Claims
1.0 (28.02.2021)	Summary and Recommendation
1.5 (01.03.2021)	KYC

2. About the Project and Company

Company address:

NEXT.exchange B.V.
Flight Forum 810
5657 DV Eindhoven
Netherlands

Website: https://nextchain.dev
Blog: https://next-chain.medium.com
Twitter: https://twitter.com/NextChainGlobal
Telegram Channel: https://t.me/nextchainNews
Telegram Group: https://t.me/nextchain
GitHub: https://github.com/NextExchange
Reddit: https://www.reddit.com/r/nextchain
Discord: https://discord.gg/gZjhrG3KcT
Disqus: https://nextchain.disqus.com

2.1 Project Overview

NEXT.chain is the next-generation blockchain - fast and easy asset creation, seamless asset tokenization, and instant trading at lightning speeds at near-zero cost. The complete blockchain ecosystem itself consists of:

- NEXT.exchange: a hybrid cryptocurrency exchange combining decentralized and centralized exchange functionality on top of NEXT.chain.
- NEXT.chain: a hybrid Proof-of-Stake and Proof-of-Work Blockchain with support for asset creation.
- NEXT.genesis: a token launch pad and cryptocurrency crowdfunding platform with support for dynamic pricing and a wide range of payment options.
- NEXT.bridge: a bridge for moving digital assets between multiple blockchains.
- NEXT.swap: a simplified version of NEXT.exchange to swap between assets and fiat.
- NEXT: the ecosystem’s digital utility asset that sees a variety of uses on NEXT.exchange, NEXT.chain, and NEXT.genesis

The platform provides individual Blockchain wallet addresses for each user’s assets, the ability to store, deposit, withdraw, and trade cryptocurrencies directly against fiat currencies, giving IBAN/SEPA accounts and connect e-commerce shops all over the world directly on NEXT.chain.

The goal is to create a complete ecosystem where it 'easy and safe' to trade cryptocurrencies on a convenient, user-friendly interface that connects social, company information, and in-depth analyses to make it as simple as possible for our users to join the digital economy.

2.2 KYC

Date of verification: 01.03.2021 14:00 CET via Zoom

Name	Position	Passport	Verified by
Christiaan van Steenbergen Full Name: Catharinus Adrianus Wilhelmus Van Steenbergen	CEO	✅	Yannik Heinze (CEO Chainsulting)
Johannes Tadeusz Tarquin Bucur	CCO	✅	Yannik Heinze (CEO Chainsulting)

Connected Companies

Company	Registration No.	Address	Since	Verified by
Anaconda Beheer B.V.	814169648	Flight Forum 810, 5657DV Eindhoven	25-08-2005	Yannik Heinze (CEO Chainsulting)
NEXT.exchange B.V.	859805694	Flight Forum 810, 5657DV Eindhoven	06-03-2019	Yannik Heinze (CEO Chainsulting)

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 Chainsulting 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 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 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.

4.2 Used Code from other Frameworks/Smart Contracts

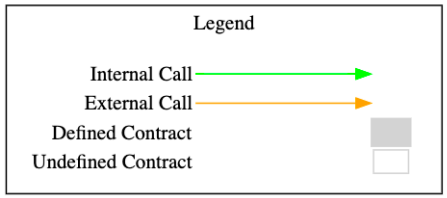
1. SafeMath.sol (0.8.0)
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/math/SafeMath.sol
2. IERC1363Spender.sol
https://github.com/vittominacori/erc1363-payable-token/blob/master/contracts/token/ERC1363/IERC1363Spender.sol
3. IERC1363Receiver.sol
https://github.com/vittominacori/erc1363-payable-token/blob/master/contracts/token/ERC1363/IERC1363Receiver.sol
4. ERC165
https://github.com/OpenZeppelin/openzeppelin-contracts/tree/master/contracts/mocks/ERC165
5. ERC1363.sol
https://github.com/vittominacori/erc1363-payable-token/blob/master/contracts/token/ERC1363/ERC1363.sol

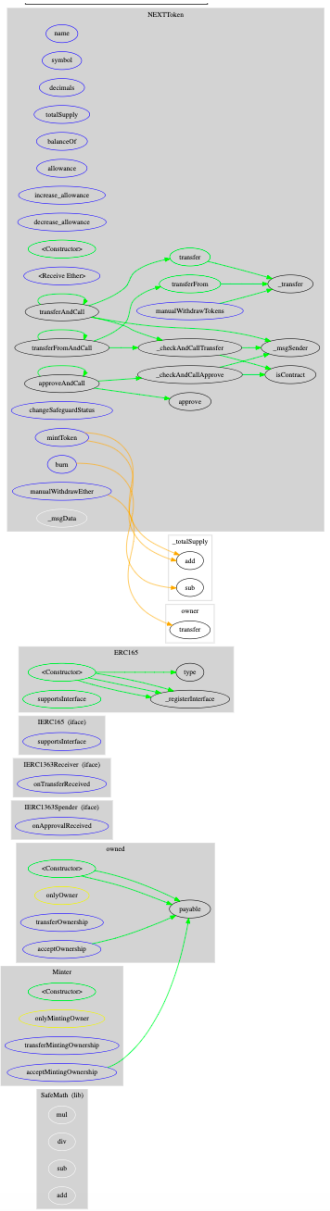
4.3 Tested Contract Files

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

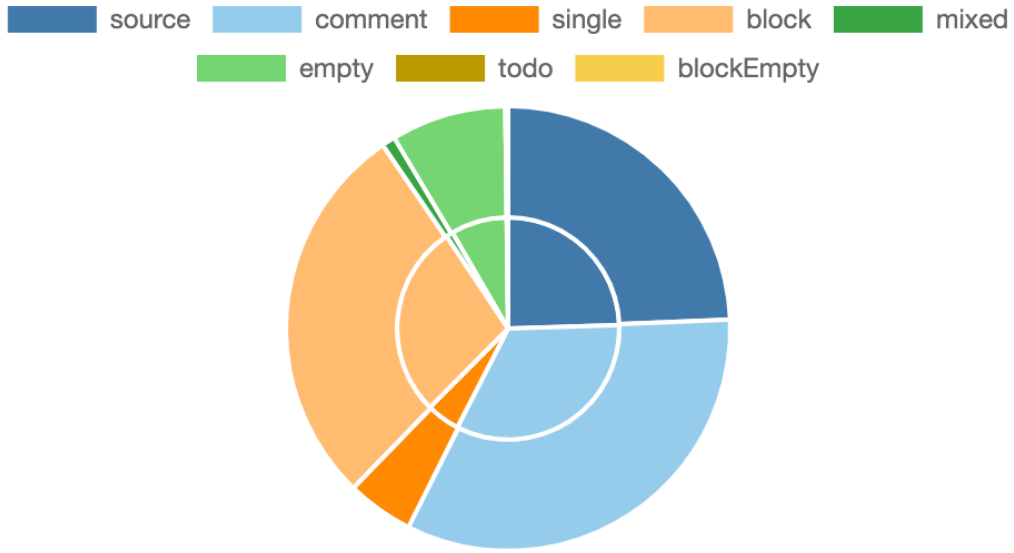
File	Fingerprint (MD5)
NEXTToken.sol	0853502c818be78bfe15bebb18eeb8ee

4.4 Metrics / CallGraph





4.5 Metrics / Source Lines



4.6 Metrics / Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
`0.8.1`		`yes`	`yes` (1 asm blocks)	

Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
`yes`					

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	NEXTToken.sol	5	3	663	648	246	339	198	
	Totals	**5**	**3**	**663**	**648**	**246**	**339**	**198**	

Legend:

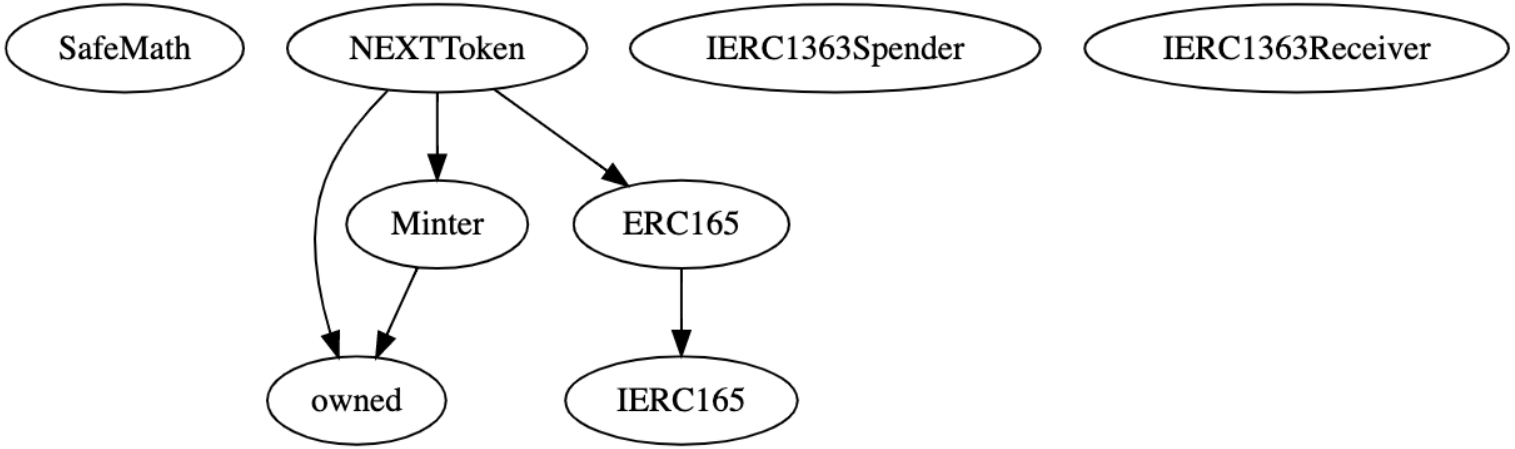
- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments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

5. Scope of Work

The Next Exchange team provided us with the files that needs to be tested. The scope of the audit is the ERC20 Token contract (NEXT)

Verify claims:

1. ERC-20 Token implementation
2. Deployer cannot mint any new tokens.
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract



5.1 Manual and Automated Vulnerability Test

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

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

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

LOW ISSUES
During the audit, Chainsulting's experts found **no Low issues** in the code of the smart contract.

5.2. SWC Attacks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✅
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✅
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✅
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✅
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✅
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✅
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✅
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✅

ID	Title	Relationships	Test Result
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✅
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✅
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✅
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✅
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✅
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✅

ID	Title	Relationships	Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✅
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✅
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✅
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✅
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✅
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✅
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✅
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✅

ID	Title	Relationships	Test Result
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	✅
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✅
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✅
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅

5.3 Verify Claims

5.2.1 ERC-20 Token implementation ✅
Status: tested and verified

5.2.2 Deployer cannot mint any new tokens. ✅
Status: tested and verified
Initial Supply: 8,000,000 (8M)
Max Supply: 30,300,000 (30,3M)
Code: Ln 443 – 448

```
function mintToken(address target, uint256 mintedAmount) onlyMintingOwner external {
        require(_totalSupply.add(mintedAmount) <= maxSupply, "Cannot Mint more than maximum supply");
        _balanceOf[target] = _balanceOf[target].add(mintedAmount);
        _totalSupply = _totalSupply.add(mintedAmount);
        emit Transfer(address(0), target, mintedAmount);
    }
```

The NEXT ERC20 Token works as a bridge between the NEXT Chain and Ethereum Chain, that's why the mint function is needed.

5.2.3 Deployer cannot burn or lock user funds ✅
Status: tested and verified
Code: Ln 428 – 436
Needs to be added as the contracts acts like a bridge between Next Chain and Ethereum Chain

```
    function burn(uint256 _value) external returns (bool success) {
        require(!safeguard);
        //checking of enough token balance is done by SafeMath
        _balanceOf[msg.sender] = _balanceOf[msg.sender].sub(_value);  // Subtract from the sender
        _totalSupply = _totalSupply.sub(_value);                      // Updates totalSupply
        emit Burn(msg.sender, _value);
        emit Transfer(msg.sender, address(0), _value);
        return true;
    }
```

Event is included into the smart contract, but no function freezeAccount.

```
    // This generates a public event for frozen (blacklisting) accounts
    event FrozenAccounts(address target, bool frozen);
```

5.2.4 Deployer cannot pause the contract ✅
Status: tested and verified

6. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The final debrief took place on the March 01, 2021. The overall code quality of the project is very good, not overloaded with unnecessary functions, these is greatly benefiting the security of the contract. It correctly implemented widely-used and reviewed contracts from OpenZeppelin and for safe mathematical operations and as well used the newest solidity compiler version.

The main goal of the audit was to verify the claims regarding the security of the smart contract and the claims inside the scope of work. During the audit, no issues were found after the manual and automated security testing.

7. Deployed Smart Contract

VERIFIED

NEXT Token
0x377d552914e7a104bc22b4f3b6268ddc69615be7