



Unicrypt

ENMTv2

SMART CONTRACT AUDIT

11.01.2023

Made in Germany by Chainsulting.de



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 Used Code from other Frameworks/Smart Contracts	11
5.3 CallGraph.....	12
5.4 Inheritance Graph	14
5.5 Source Lines & Risk.....	15
5.6 Capabilities	16
5.7 Source Unites in Scope	17
6. Scope of Work.....	20
6.1 Findings Overview	21
6.2 Manual and Automated Vulnerability Test.....	23
6.2.1 Wrong Implementation Of Diamond Patterns	23
6.2.2 Wrong Implementation Of Ownable For Diamond Patterns.....	25
6.2.3 High Gas Consumption For Token Creation.....	25
6.2.4 Bad Test Coverage	26
6.2.5 Missing Implementations For Diamond Standard	28

6.2.6 Missing Batch Function For Adding Facets And Function Selectors	28
6.2.7 Missing Remove And Replace Facet Functions	29
6.2.8 Redundant Byte Code	30
6.2.9 Unnecessary If Statement.....	30
6.2.10 Missing Value Validation	31
6.2.11 Duplicated Storing Of Variables.....	32
6.2.12 Missing Inheritance	33
6.2.13 Missing Natspec Documentation	33
6.2.14 Wrong Function Returning Parameter	34
6.2.15 Redundant Getter Functions	35
6.2.16 Unexplicit State Variable Visibility	36
6.2.17 Unexplicit Variable Declaration	36
6.2.18 Misleading Contract Name	37
6.2.19 Naming Conventions Violation.....	38
6.3 SWC Attacks	39
6.4 Verify Claims	43
6.5 Unit Tests	44
7. Executive Summary.....	59
8. About the Auditor	60

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 SDD Tech OÜ. 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 (19.12.2022)	Layout
0.4 (26.12.2022)	Automated Security Testing Manual Security Testing
0.5 (02.01.2023)	Verify Claims and Test Deployment
0.6 (03.01.2023)	Testing SWC Checks
0.9 (05.01.2023)	Summary and Recommendation
1.0 (07.01.2023)	Final document
1.1 (11.01.2023)	Re-check 936ab7311e04eb294802de68cb7002b5a41fe754

2. About the Project and Company

Company address:

SDD Tech OÜ
Mustamäe tee 6b
Tallinn Harjumaa 10616

Website: <https://unicrypt.network>

Twitter: https://twitter.com/UNCX_token

Telegram: https://t.me/uncx_token

Medium: <https://unicrypt.medium.com>



2.1 Project Overview

UniCrypt is a decentralized services provider which offers several ways for DeFi projects to build community trust and keep users safe. Famously, UniCrypt created the first-ever liquidity locking smart contracts for Uniswap on Ethereum, known as Proof-of-Liquidity or POL. From there the project continued to develop new features, combining liquidity locking with a decentralized launchpad.

Liquidity Lockers: these are smart contracts that enable teams to publicly lock liquidity on Uniswap or other AMMs for a predetermined period. Essentially, it's a guarantee to investors that the project developers can't drain the pool of all the funds. A key innovation is UniCrypt's lockers will be able to migrate liquidity to Uniswap V3 when the time comes.

FaaS: This is a yield farming-as-a-service protocol that enables the creation of a farm for any token. Launch a farm in a couple clicks using the UI, all automatic with no coding necessary.

Launchpad: Perhaps the most interesting service, a 100% decentralized and automated presale platform that is connected to the liquidity lockers. Once the presale ends a portion of the raised funds (between 30% to 100%) will create the DEX pair on a supported AMM and the liquidity will be locked.

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.

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)
./LPWallet.sol	850e381c571fc16dd7d9517ccf4a728a
./TaxHelperUniswapV2.sol	83f87a608124b7ac6a22e4f7ebf22975
./facets/Settings.sol	39d5b58f01718149814c992c2be367c8
./facets/Wallets.sol	3c29a0647857f2b6e543a7f230b4f84a
./facets/Lossless.sol	531e482d5ed2d4ff988d729a8fd66c1b
./facets/Multicall.sol	0ced55cac2baf45b662084edf94831b9
./facets/Storage.sol	4b9024f47e7f688913e9cb47882751c6
./facets/Constructor.sol	e958f1a349ded4e1eccf7c27ba8a38a2
./facets/Tax.sol	6bbcb81c2f66eabc731d91b9726e6432
./facets/AntiBot.sol	6bc20fcce43d13a33477773a3097f397
./MintFactory.sol)	3b88bd21528d647fb442d0bfecce46a6
./MintGenerator.sol	741efe58f4a6f701a40d7ad4506f7870
./FeeHelper.sol	5e72dba1eefc63cee13dcac133200bdf
./TaxToken.sol	2db43f0d9349f1e8c2aafb04f8ffd206
./FacetHelper.sol	b3a64385e4b3f6a430230e01f7b37e3c
./BuyBackWallet.sol	94268e7dd3b8711584cc4c6f54920198
./interfaces/IERC20.sol	402f41cf15481fc66a14e9716fad01ff

./interfaces/ITaxHelper.sol	e17ff9b27ded912f9fe1e93e0d97ce7c
./interfaces/IWallets.sol	d3bc6f6248306fbb134d9d1bee36635e
./interfaces/IUniswapV2Router01.sol	3526c6d8e16194c21ba5df9314876f13
./interfaces/IUniswapV2Router02.sol	0b40738178248d5a550bcb99b5451260
./interfaces/IMintFactory.sol	a38b7dac134bb3a1392f342cd9e2ad88
./interfaces/ITaxToken.sol	cec77575c551cd75a4aeef2730b69cb
./interfaces/IBuyBackWallet.sol	42d1f350106a5312280f6d48e4ccadc3
./interfaces/IUniswapV2Factory.sol	c1b36b46a276aed7e0954e0036e050c8
./interfaces/ILosslessController.sol	5c50e88b5c533f77d9d062a1ee337a45
./interfaces/IUniswapV2Pair.sol	1d4d8e255a202a54b00503590bd11e30
./interfaces/IFacetHelper.sol	18ccfdcad33bac105cd875e7ae3aef00
./interfaces/ISettings.sol	654a35be2f37489d1c4cf6156206f2f5
./interfaces/IFeeHelper.sol	94fbfea15e83a91280c8c5fd5600fb6c
./interfaces/ILPWallet.sol	0678fea99b407c9a7655e64d3862b55f

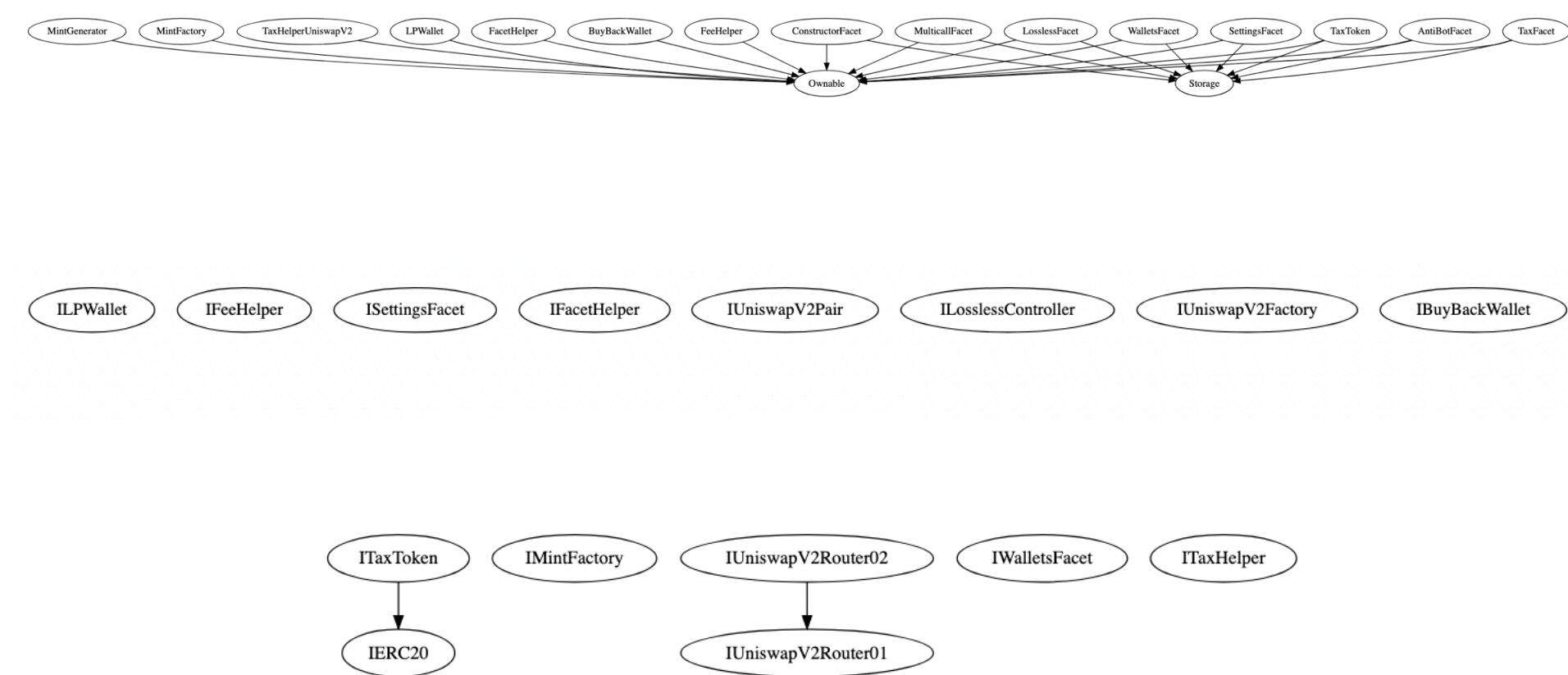
5.2 Used Code from other Frameworks/Smart Contracts (direct imports)

Dependency / Import Path	Source
@openzeppelin/contracts/utils/Context.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.0.0/contracts/utils/Context.sol
@openzeppelin/contracts/token/ERC20/ERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.0.0/contracts/token/ERC20/ERC20.sol
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.0.0/contracts/access/Ownable.sol
@uniswap/v2-core	https://github.com/Uniswap/v2-core
@uniswap/v2-periphery	https://github.com/Uniswap/v2-periphery

5.3 CallGraph

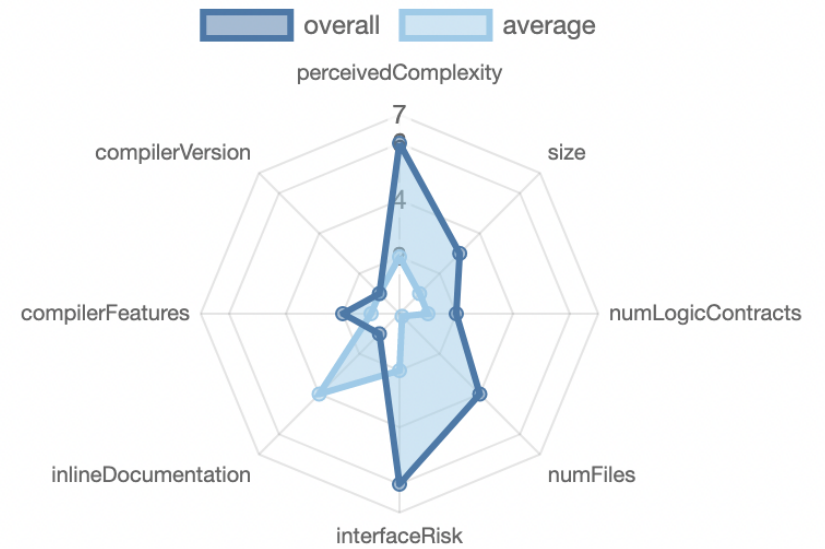
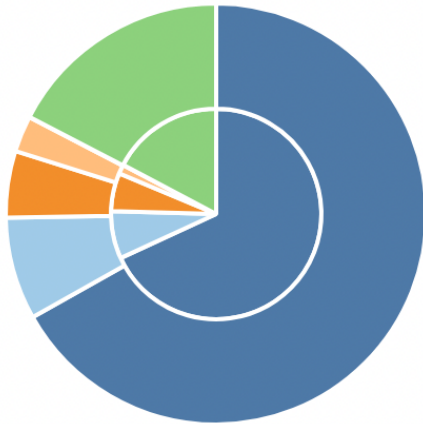


5.4 Inheritance Graph













5.5 Source Lines & Risk

source comment single block mixed
empty todo blockEmpty





5.6 Capabilities














Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts	
0.8.17 >=0.5.0 >=0.6.2		yes	yes (1 asm blocks)		
 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRrecover	 New/Create/Create2
yes		yes	yes		yes → NewContract:TaxToken → NewContract:BuyBackWallet → NewContract:LPWallet











Exposed Functions















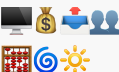
This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

 Public	 Payable				
283	9				
External	Internal	Private	Pure	View	
185	194	12	11	117	

5.7 Source Unites in Scope

Type	File	Logic Contract s	Interface s	Lin es	nLin es	nSL OC	Comment Lines	Compl ex. Score	Capabiliti es
	contracts/FacetHelper.sol	1	_____	151	151	108	11	81	_____
	contracts/BuyBackWallet.sol	1	_____	66	66	46	3	37	💰
	contracts/FeeHelper.sol	1	_____	55	55	38	3	22	_____
	contracts/TaxToken.sol	1	_____	373	373	289	30	301	💻👥
	contracts/MintGenerator.sol	1	_____	47	45	22	8	34	💰🔗🎯
	contracts/MintFactory.sol	1	_____	181	181	109	35	84	_____
	contracts/TaxHelperUniswapV2.sol	1	_____	119	119	95	6	108	💰🔗
	contracts/LPWallet.sol	1	_____	77	77	54	3	46	💰🔗
	contracts/facets/AntiBot.sol	1	_____	170	170	132	7	112	_____
	contracts/facets/Tax.sol	1	_____	342	342	271	23	176	_____
	contracts/facets/Constructor.sol	1	_____	161	161	141	8	95	_____
	contracts/facets/Storage.sol	1	_____	189	189	135	7	52	_____
	contracts/facets/Multicall.sol	1	_____	172	172	134	12	71	_____

Type	File	Logic Contract s	Interface s	Lin es	nLin es	nSL OC	Comment Lines	Compl ex. Score	Capabiliti es
	contracts/interfaces/ILPWall et.sol		1	21	10	3	3	13	
	contracts/interfaces/IFeeHel per.sol		1	22	9	3	3	15	
	contracts/interfaces/ISetting s.sol		1	13	10	3	3	7	
	contracts/interfaces/IFacetH elper.sol		1	46	10	3	3	37	
	contracts/interfaces/IUniswa pV2Pair.sol		1	52	7	5		55	
	contracts/interfaces/ILossles sController.sol		1	24	7	3	1	25	
	contracts/interfaces/IUniswa pV2Factory.sol		1	17	6	4		17	
	contracts/interfaces/IBuyBac kWallet.sol		1	19	10	3	3	11	
	contracts/interfaces/ITaxTok en.sol		1	19	12	4	3	9	
	contracts/interfaces/IMintFa ctory.sol		1	43	16	8	3	29	

Type	File	Logic Contract s	Interface s	Lin es	nLin es	nSL OC	Comment Lines	Compl ex. Score	Capabiliti es
	contracts/interfaces/IUniswapV2Router02.sol	_____	1	44	6	4	_____	16	
	contracts/interfaces/IUniswapV2Router01.sol	_____	1	95	4	3	_____	48	
	contracts/interfaces/IWallets.sol	_____	1	16	10	3	3	9	_____
	contracts/interfaces/ITaxHelper.sol	_____	1	23	10	3	3	9	_____
	contracts/interfaces/IERC20.sol	_____	1	79	28	17	58	13	
	contracts/facets/Lossless.sol	1	_____	67	67	49	4	43	
	contracts/facets/Wallets.sol	1	_____	43	43	27	3	49	
	contracts/facets/Settings.sol	1	_____	212	212	177	4	132	_____
	Totals	16	15	2958	2578	1896	253	1756	

- **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

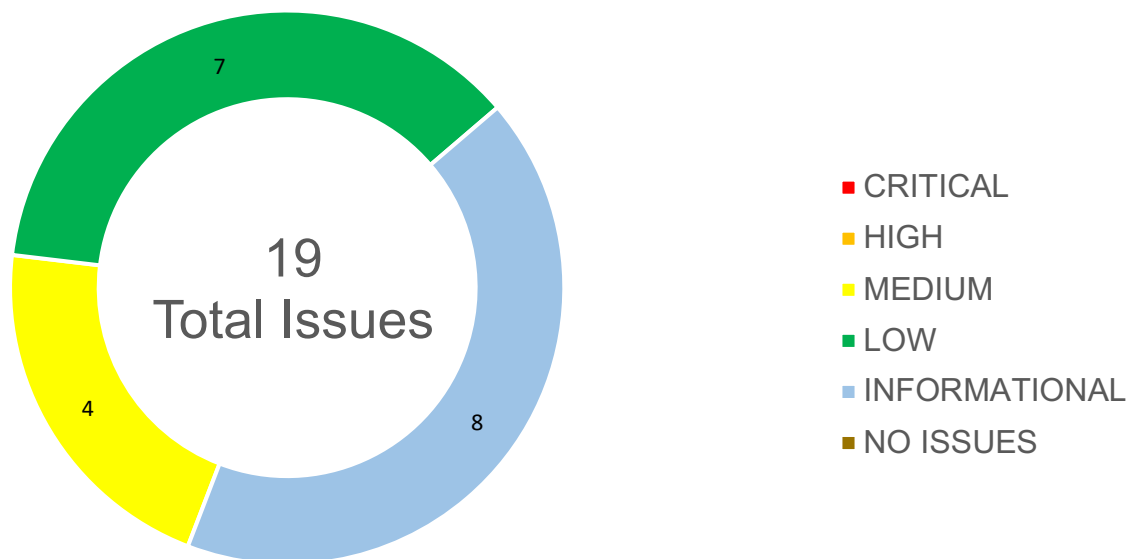
The Unicrypt Team provided us with the files that needs to be tested. The scope of the audit is the ENMTv2 contract.

The team put forward the following assumptions regarding the security, usage of the contracts:

- The contract is using the ERC-20 token standard
- The minting of token is working as expected
- The ENMT contract owner cannot control minted tokens
- The smart contract is coded according to the newest standards and in a secure way

The main goal of this audit was to verify these claims. 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	Wrong Implementation Of Diamond Patterns	MEDIUM	FIXED
6.2.2	Wrong Implementation Of Ownable For Diamond Patterns	MEDIUM	FIXED
6.2.3	High Gas Consumption For Token Creation	MEDIUM	FIXED
6.2.4	Bad Test Coverage	MEDIUM	FIXED
6.2.5	Missing Implementations For Diamond Standard	LOW	ACKNOWLEDGED
6.2.6	Missing Batch Function For Adding Facets And Function Selectors	LOW	ACKNOWLEDGED
6.2.7	Missing Remove And Replace Facet Functions	LOW	ACKNOWLEDGED
6.2.8	Redundant Byte Code	LOW	ACKNOWLEDGED
6.2.9	Unnecessary If Statement	LOW	FIXED

6.2.10	Missing Value Validation	LOW	ACKNOWLEDGED
6.2.11	Duplicated Storing Of Variables	LOW	ACKNOWLEDGED
6.2.12	Missing Inheritance	INFORMATIONAL	ACKNOWLEDGED
6.2.13	Missing Natspec Documentation	INFORMATIONAL	ACKNOWLEDGED
6.2.14	Wrong Function Returning Parameter	INFORMATIONAL	FIXED
6.2.15	Redundant Getter Functions	INFORMATIONAL	FIXED
6.2.16	Unexplicit State Variable Visibility	INFORMATIONAL	ACKNOWLEDGED
6.2.17	Unexplicit Variable Declaration	INFORMATIONAL	ACKNOWLEDGED
6.2.18	Misleading Contract Name	INFORMATIONAL	ACKNOWLEDGED
6.2.19	Naming Conventions Violation	INFORMATIONAL	ACKNOWLEDGED

6.2 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 **4 Medium issues** in the code of the smart contract

6.2.1 Wrong Implementation Of Diamond Patterns

Severity: MEDUM

Status: FIXED

Code: CWE-1068

File(s) affected: TaxToken.sol, Antibot.sol, Constructor.sol, Lossless.sol, Multicall.sol, Settings.sol, Storage.sol, Tax.sol, Wallets.sol

Update: The patterns are implemented as intended by the developer

Attack / Description	According to the EIP-2535 specification of diamond patterns, only the diamond contract is stateful and stores data. Facets are stateless and defining functions to be called from the diamond contract. In the current implementation, all facets are statefull by defining their own state and functions to modify this state. Some functions of the facets are directly called by the diamond contract by using delegateCall directly in the code. This is not according to the EIP-2535. The total storage of the diamond contract is defined as a whole in every facet contract. Facets should be defining just a small part of the total functionality of the diamond and thus only need to have access to a small part of the variables. This storage layout produces a massive overhead of storage.
Code	NA

Result/Recommendation	<p>It is highly recommended to go through the EIP-2535 specification and especially inspect the storage patterns proposed in the storage section. For a correct implementation of diamond patterns, it is highly recommended to use one of the three templates proposed by Nick Mudge (mudgen). The proposed architecture can simply be forked and extended with libraries and facet contracts.</p> <p>It is recommended to define storage and reused functions in libraries and use these libraries in the different facets, due to the needs of its functionality. It is not necessary to have all variables visible in all facets.</p> <p>It is highly recommended to replace the delegate calls with abi encoding with library call. The abi encoding of function signatures is very error-prone and changes won't be reflected by the compiler. Delegate calls should only be used inside the fallback function to delegate the desired function calls to the correct facet implementing this function.</p> <p><u>Elements of the diamond:</u></p> <ol style="list-style-type: none"> 1. Diamond contract <ul style="list-style-type: none"> - the only entry point to be externally called by users - is stateful (holds all storage variables defined in libraries) - holds immutable functions - holds references of function selectors to the implementing facets - implements a fallback function to delegate function calls to facets 2. Libraries <ul style="list-style-type: none"> - should define storage layout and variables - holding functions to interact with the storage (read/write/) and additional internal functions 3. Facets <ul style="list-style-type: none"> - defining external function called by the diamond contract via delegate call from the fallback function (i.e. transfer, balanceOf, ...) - using libraries to access storage of the diamond contract correctly - can be replaced easily with new facets by resetting references in diamond contract

	Links: EIP-2535: https://eips.ethereum.org/EIPS/eip-2535 Diamond Templates: https://github.com/mudgen?tab=repositories
--	---

6.2.2 Wrong Implementation Of Ownable For Diamond Patterns

Severity: MEDUM

Status: FIXED

Code: CWE-1068

File(s) affected: TaxToken.sol, Antibot.sol, Constructor.sol, Lossless.sol, Multicall.sol, Settings.sol, Tax.sol, Wallets.sol

Update: The patterns are implemented as intended by the developer

Attack / Description	In the current implementation, all facets of the diamond are inheriting from Ownable. This means every facet implements its own owner and functions to transfer ownership. In a diamond pattern, there should only be one owner for each diamond contract known by the facets.
Code	NA
Result/Recommendation	Use a library to define one owner for the diamond contract and use this library for access control in each facet. In this way there is only one consistent owner for each diamond with all facets. It is highly recommended to use an ownership facet like in mudgen templates to manage ownership of an diamond. Link: https://github.com/mudgen/diamond-3-hardhat/blob/main/contracts/facets/OwnershipFacet.sol

6.2.3 High Gas Consumption For Token Creation

Severity: MEDIUM

Status: FIXED

Code: NA

File(s) affected: MintGenerator.sol

Update: <https://github.com/Boka44/enmt-v2/commit/936ab7311e04eb294802de68cb7002b5a41fe754>

Attack / Description	In the current implementation, the compiler optimizer is configured to one run for the MintGenerator.sol. This is probably done to reduce the contracts byte code size but increases the used gas on each function call significantly. The token generation consumes over 9 million gas, which is around 160 USD at the time of writing. This can increase drastically if the gas and token price is rising. At all-time high conditions, the price would be around 6500 USD per function call to create a new token.
Code	NA
Result/Recommendation	It is recommended to use a higher runs parameter for the optimizer to lower the gas costs for each function call. To reduce the byte code of the MintGenerator it is recommended to reduce the byte code of the TaxToken contract by implementing the Diamond pattern properly.

6.2.4 Bad Test Coverage

Severity: MEDIUM

Status: FIXED

Code: NA

File(s) affected: ALL

Update: <https://github.com/Boka44/enmt-v2/commit/936ab7311e04eb294802de68cb7002b5a41fe754>

Attack / Description	In the current implementation, several functions are never tested and other functions are rarely tested.
Code	List of totally untested function: 1. taxToken: <ul style="list-style-type: none">- transferOutBlacklistedFunds- tokenFromReflection- increaseAllowance- decreaseAllowance- burn

	<p>2. lossless facet:</p> <ul style="list-style-type: none">- all functions <p>3. Antibot:</p> <ul style="list-style-type: none">- setIncrement- setEndDate- setInitialMaxHold- updateAntiBot- removeMaxBalanceWhitelistedAddress- updateMaxBalanceWhitelistBatch- removeSwapWhitelistedAddress- updateSwapWhitelistBatch- setSwapWhitelistEndDate <p>4. Settings:</p> <ul style="list-style-type: none">- addLPToken- removeLPToken- removeBlacklistedAddress- updateBlacklistBatch- updateTaxFees- updateTransactionTaxAddress- updatePairAddress- updateTaxHelperIndex <p>5. Tax facet:</p> <ul style="list-style-type: none">- reflect- reflectionFromToken- tokenFromReflection- includeAccount <p>6. FacetHelper:</p> <ul style="list-style-type: none">- resetFacetStorage <p>7. MintFactory:</p> <ul style="list-style-type: none">- updateTaxHelper
--	--

Result/Recommendation	It is recommended to enhance the test coverage to reach nearly 100%. Tests are crucial to validate that the contracts are behaving and functioning as expected. To check the total test coverage, it is recommended to use solidity-coverage hardhat plugin and test every branch of every function call with happy and sad path testing.
------------------------------	---

LOW ISSUES

During the audit, Chainsulting's experts found **7 Low issues** in the code of the smart contract

6.2.5 Missing Implementations For Diamond Standard

Severity: LOW

Status: ACKNOWLEDGED

Code: CWE-1068

File(s) affected: TaxToken.sol

Attack / Description	According to the EIP-2535 specification "all diamonds must implement the IDiamond interface" and "diamonds must emit DiamondCut event" on functions modification. Additionally, "diamonds must support inspecting and functions by implementing the IdiamondLoupe interface" to enable correct display of all diamond related facet contract and its functions. The current implementation is missing these implementation standards.
Code	NA
Result/Recommendation	It is recommended to implement all specifications according to the EIP-2535 standard defined in section "Implementation Points". This is important to enable support of other DApps and tools to inspect and interact with the diamond contract.

6.2.6 Missing Batch Function For Adding Facets And Function Selectors

Severity: LOW

Status: ACKNOWLEDGED



Code: CWE-1068

File(s) affected: FacetHelper.sol

Attack / Description	The current implementation is missing a batch add function to add function selectors for a facet. For every function of every facet the addFacet and addSelector function has to be called once. This leads to many function calls on initialization.
Code	NA
Result/Recommendation	It is highly recommended to use the diamondCut function proposed by mudgen to add, remove and replace multiple facets and selectors at once. Source: https://github.com/mudgen/diamond-3-hardhat/blob/main/contracts/libraries/LibDiamond.sol

6.2.7 Missing Remove And Replace Facet Functions

Severity: LOW

Status: ACKNOWLEDGED

Code: CWE-1068

File(s) affected: FacetHelper.sol

Attack / Description	The current implementation is missing a remove and replace function for facet functions. If one facet should be removed, all facets must be reset and for every other function the addFacet and addSelector has to be called. This results in many function calls for just one removal or replacement of a function selector.
Code	NA
Result/Recommendation	It is highly recommended to use the diamondCut function proposed by mudgen to add, remove, and replace facets and selectors.

	Source: https://github.com/mudgen/diamond-3-hardhat/blob/main/contracts/libraries/LibDiamond.sol
--	---

6.2.8 Redundant Byte Code

Severity: LOW

Status: ACKNOWLEDGED

Code: CWE-1041

File(s) affected: TaxToken.sol, Tax.sol, Constructor.sol, Settings.sol, Multicall.sol

Attack / Description	Some functions are defined multiple times in different facets with exact the same byte code. This produces redundant byte code for the diamond contract.
Code	<p>checkMaxTax: Constructor.sol, Multicall.sol, Settings.sol</p> <p>addLPToken: Constructor.sol, Settings.sol</p> <p>paused: TaxToken.sol, Settings.sol, TaxFacet</p> <p>isBlacklisted: TaxToken.sol, Settings.sol, Tax.sol</p>
Result/Recommendation	It is highly recommended to create libraries for recurring functions and use them in the affected facets. Provide only one external function per diamond contract to be called.

6.2.9 Unnecessary If Statement

Severity: LOW

Status: ACKNOWLEDGED

Code: NA

File(s) affected: TaxHelperUniswapV2.sol, BuyBackWallet.sol, LPWallet.sol

Update: <https://github.com/Boka44/enmt-v2/commit/936ab7311e04eb294802de68cb7002b5a41fe754>

Attack / Description	In the current implementation are several if statements returning true if the statement is true. This is redundant and consumes more gas than returning just the statement.
Code	Line 34 – 39 (BuyBackWallet.sol) <pre>function checkBuyBackTrigger() public view returns (bool) { if(address(this).balance > threshold) { return true; } return false; }</pre> TaxHelperUniswapV2.LpTokenHasReserves
Result/Recommendation	It is recommended to remove the if and just return the statement to reduce byte code.

6.2.10 Missing Value Validation

Severity: LOW

Status: ACKNOWLEDGED

Code: CWE-20

File(s) affected: NA

Attack / Description	Several functions lack value safety checks. Therefore, only values that are consistent with the logic of the contract should be permitted. Missing address validation checks could lead to contract deployment without properly set addresses.
Code	i.e.: Line 30 – 33 (Settings.sol) <pre>function addLPToken(address _newLPToken) public onlyOwner { lpTokens[_newLPToken] = true; emit AddedLPToken(_newLPToken); }</pre>

	<p>Line 48 – 50 (FacetHelper.sol)</p> <pre>function setFeeAddress(address payable _feeAddress) external onlyOwner { SETTINGS.FEE_ADDRESS = _feeAddress; }</pre> <p>Line 44 – 46 (FacetHelper.sol)</p> <pre>function setFee(uint _fee) external onlyOwner { SETTINGS.FEE = _fee; }</pre>
Result/Recommendation	<p>It is recommended to check address values for correctness. This can be done in first stage to exclude zero address in a require statement. In second stage to check if an address is a contract. And most specific for contracts if an address implements a specified interface (EIP-165). Additionally, it is recommended to allow only variable values consistent with the contract logic (i.e. fee values not greater than denominator).</p>

6.2.11 Duplicated Storing Of Variables

Severity: LOW

Status: ACKNOWLEDGED

Code: NA

File(s) affected: MintFactory.sol

Attack / Description	In the current implementation the MinFactory stores the referencing addresses for taxHelpers duplicated in a mapping and array. This produces storage overhead.
Code	<p>Line 27 – 28 (MintFactory.sol)</p> <pre>mapping(uint => TaxHelper) public taxHelpersData; address[] public taxHelpers;</pre>

Result/Recommendation	It is recommended to only use the the mapping and an uint counter for tracking the number of registered taxHelpers to track the indices. This reduces storage usage and gas costs.
------------------------------	--

INFORMATIONAL ISSUES

During the audit, Chainsulting's experts found **8 Informational issues** in the code of the smart contract

6.2.12 Missing Inheritance

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: NA

File(s) affected: LPWallet.sol, BuyBackWallet.sol

Attack / Description	The LPWallet implements exactly the same functions as BuyBackWallet with some additional functions.
Code	NA
Result/Recommendation	It is recommended to used inheritance. LPWallet should inherit from BuyBackWallet to be consistent with the function implementations.

6.2.13 Missing Natspec Documentation

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: CWE-1059

File(s) affected: ALL

Attack / Description	Solidity contracts can use a special form of comments to provide rich documentation for functions, return variables and more. This special form is named the Ethereum Natural Language Specification Format (NatSpec).
Code	NA
Result/Recommendation	It is recommended to include natspec documentation and follow the doxygen style including @author, @title, @notice, @dev, @param, @return and make it easier to review and understand your smart contract.

6.2.14 Wrong Function Returning Parameter

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: NA

File(s) affected: AntiBot.sol

Update: <https://github.com/Boka44/enmt-v2/commit/936ab7311e04eb294802de68cb7002b5a41fe754>

Attack / Description	The current implementation of the isMaxBalanceWhitelisted function returns a value of the swapWhitelist variable.
Code	Line 92 – 94 (AntiBot.sol) <pre>function isMaxBalanceWhitelisted(address _address) public view returns (bool) { return swapWhitelist[_address]; }</pre>
Result/Recommendation	It is recommended to return the variable <i>maxBalanceWhitelist</i> , which is indicated by the function name.

6.2.15 Redundant Getter Functions

Severity: INFORMATIONAL

Status: FIXED

Code: NA

File(s) affected: LPWallet.sol, BuyBackWallet.sol, MintFactory.sol

Update: <https://github.com/Boka44/enmt-v2/commit/936ab7311e04eb294802de68cb7002b5a41fe754>

Attack / Description	The current implementation has getter functions for public state variables. Public state variables automatically generate getter functions.
Code	<p>Line 20 (BuyBackWallet.sol) <code>uint256 public threshold;</code></p> <p>Line 41 – 43 (BuyBackWallet.sol) <code>function getBalance() public view returns (uint256) { return address(this).balance; }</code></p> <p>Line 60 – 62 (BuyBackWallet.sol) <code>function getThreshold() external view returns (uint256) { return threshold; }</code></p> <p>Factory: getTaxHelperAddress, getTaxHelpersDataByIndex, getTokenByOwnerAtIndex, getFacetHelper, getFeeHelper, getLosslessController</p>
Result/Recommendation	It is recommended to remove the redundant getter functions or declare the state variables as private to reduce byte code size.

6.2.16 Unexplicit State Variable Visibility

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: NA

File(s) affected: TaxHelperUniswapV2.sol, Storage.sol

Attack / Description	In the current implementation several state variables are leaking explicit visibility. Implicitly these variables are defined as public.
Code	Line 23 – 25 (TaxHelperUniswapV2.sol) IUniswapV2Router02 router; IUniswapV2Factory factory; IMintFactory mintFactory; Line 62 (Storage.sol) <code>uint256 constant MAX = ~uint256(0);</code>
Result/Recommendation	Add explicit visibility types to all state variables to ensure availability and desired access control for the lowest possible gas costs. Ref.: https://docs.soliditylang.org/en/v0.8.11/contracts.html#visibility-and-getters

6.2.17 Unexplicit Variable Declaration

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: NA

File(s) affected: BuyBackWallet.sol, FacetHelper.sol, LPWallet.sol, MintFactory.sol, TaxToken.sol, AntiBot.sol, Constructor.sol, Multicall.sol, Settings.sol, Storage.sol

Attack / Description	In the current implementation unexplicit types of unsigned integer variables are used several times.
Code	i.e.: Line 46 (BuyBackWallet.sol) uint index = token.taxHelperIndex(); FacetHelper line 72: for(uint i = 0; i < selectorsList.length; i++)
Result/Recommendation	It is recommended to use explicit types of uint (i.e. uint256, uint128, ...) to clarify storage usage and expected size of numbers. It is also helpful to reduce storage of structs with multiple integer variables due to packing.

6.2.18 Misleading Contract Name

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: NA

File(s) affected: MintFactory.sol

Attack / Description	The MintFactory is capable of storing several different references of deployed tokens, token generators and references to other smart contracts. Its only function is to register tokens and token generators. The name factory indicates, that this contract is producing something, which is clearly not the case.
Code	MintFactory.sol

Result/Recommendation	It is recommended to use a name that reflects the functionality of the contract. It should be renamed to MintRegistry, TokenRegistry or similar to indicate that this contract registers new tokens.
------------------------------	--

6.2.19 Naming Conventions Violation

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: NA

File(s) affected: Storage.sol, FacetHelper.sol, MintGenerator.sol

Attack / Description	In the current implementation several variable names are violating best practice naming convention rules. This leads to bad code readability.
Code	<p>Storage.sol:</p> <pre> line 12: uint256 public CONTRACT_VERSION = 1; line 50: uint256 public MaxTax = 3000; line 51: uint256 public MaxTax = 3000; line 53: uint256 internal DENOMINATOR; </pre> <p>Line 12 – 19 (FacetHelper.sol)</p> <pre> struct Settings { uint256 GENERATOR_FEE; uint256 FEE; uint256 DENOMINATOR; address payable FEE_ADDRESS; } Settings public SETTINGS; </pre> <p>MintGenerator line 23 – 27:</p> <pre> uint256 public CONTRACT_VERSION = 1; IMintFactory public MINT_FACTORY; </pre>

	IFeeHelper <code>public</code> FEE_HELPER; Line 29 – 50 (TaxToken.sol) Parameters with mixed style within a struct: underscore at the beginning, underscore at the end, normal mixedCase
Result/Recommendation	It is recommended to follow the naming conventions defined in the solidity style guide. State variables should be written in mixedCase and constants with capital letters. If a variable is meant to be constant mark it with the constant or immutable keyword. https://docs.soliditylang.org/en/v0.8.17/style-guide.html

6.3 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	✓


ID	Title	Relationships	Test Result
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	✓
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	✓

ID	Title	Relationships	Test Result
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')	✓
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	✓


ID	Title	Relationships	Test Result
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	✓
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	✓

6.4 Verify Claims


6.4.1 The contract is using the ERC-20 token standard

Status: tested and verified 


6.4.2 The minting of token is working as expected

Status: tested and verified 

6.4.3 The ENMT contract owner cannot control minted tokens

Status: tested and verified 

6.4.4 The smart contract is coded according to the newest standards and in a secure way.

Status: tested and verified 

6.5 Unit Tests

Compiled 91 Solidity files successfully

Contract Name	Size (KB)	Change (KB)
Math	0.085	+0.085
UQ112x112	0.085	+0.085
SafeMath	0.085	+0.085
console	0.086	+0.086
EnumerableSet	0.086	+0.086
FullMath	0.086	+0.086
TransferHelper	0.086	+0.086
BitMath	0.086	+0.086
Babylonian	0.086	+0.086
UniswapV2Library	0.086	+0.086
FullMath	0.086	+0.086
SafeMath	0.086	+0.086
UniswapV2OracleLibrary	0.086	+0.086
UniswapV2LiquidityMathLibrary	0.086	+0.086

FixedPoint	0.295	+0.295
FeeHelper	2.722	+2.722
WETH9Unused	3.124	+3.124
UniswapV2Migrator	3.148	+3.148
WETH9	3.449	+3.449
BuyBackWallet	3.688	+3.688
ERC20	3.938	+3.938
Storage	4.081	+4.081
DeflatingERC20	4.270	+4.270
UniswapV2ERC20	4.491	+4.491
ERC20	4.491	+4.491
LPWallet	4.592	+4.592
ExampleOracleSimple	4.716	+4.716
ERC20Mock	4.894	+4.894
ERC20	4.894	+4.894
ERC20	5.049	+5.049
ERC20	5.049	+5.049
ExampleSlidingWindowOracle	5.363	+5.363

RouterEventEmitter	5.863	+5.863
ExampleSwapToPrice	6.049	+6.049
ENMT	6.169	+6.169
ExampleComputeLiquidityValue	6.506	+6.506
ExampleFlashSwap	6.654	+6.654
FacetHelper	7.709	+7.709
LosslessFacet	8.445	+8.445
TaxHelperUniswapV2	9.440	+9.440
MintFactory	9.552	+9.552
AntiBotFacet	13.670	+13.670
MulticallFacet	13.862	+13.862
UniswapV2Pair	16.176	+16.176
SettingsFacet	17.464	+17.464
ConstructorFacet	18.048	+18.048
WalletsFacet	18.295	+18.295
UniswapV2Router02	19.280	+19.280
TaxFacet	20.308	+20.308

UniswapV2Factory	20.451	+20.451	
.....
UniswapV2Router01	21.013	+21.013	
.....
TaxToken	21.070	+21.070	
.....
MintGenerator	21.120	+21.120	
-----	-----	-----	-----

```
{
  startBlock: 0,
  endDate: 48,
  increment: 50,
  initialMaxHold: 4000,
  isActive: true
}
{
  startBlock: 0,
  endDate: 48,
  increment: 50,
  initialMaxHold: 4000,
  isActive: true
}
```

AntiBot Test
0xa85233C63b9Ee964Add6F2cffe00Fd84eb32338f
Tax Sigs length: 9
Lossless Sigs length: 7
Settings Sigs length: 15
Wallets Sigs length: 4
AntiBot Sigs length: 18
Multicall Sigs length: 2
Tax Sigs Duplicates: 0
Lossless Sigs Duplicates: 0
Settings Sigs Duplicates: 0

```

Settings Sigs Duplicates: 0
AntiBot Sigs Duplicates: 0
Multicall Sigs Duplicates: 0
All Sigs Duplicates: 0
Creating Pair -----
  ✓ Should create a taxToken (142ms)
  ✓ Should set a buyBack Wallet Threshold
  ✓ Should set a LP Wallet Threshold
  ✓ Should transfer tokens to user
  ✓ Should test basic ERC20 functions
Liquidity Time
BigNumber { value: "5000000000000000000000" }
addLiquidity
Pair from factory: 0xcd905cdfdb672103d575ba3fb188ebc82cb4c7b4
factory: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512
Pair For: 0xcd905cdfdb672103d575ba3fb188ebc82cb4c7b4
factory: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512
Done
_mintFee()
true
  ✓ Should add liquidity to the pool (87ms)
  ✓ Should activate swapWhitelisting and whitelist address
false
true
  ✓ Should activate antibot (marketInit) on first buy (64ms)
Pair For: 0xcd905cdfdb672103d575ba3fb188ebc82cb4c7b4
factory: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512
Done
  ✓ Should limit buying above the max antibot hold (78ms)
Pair For: 0xcd905cdfdb672103d575ba3fb188ebc82cb4c7b4
factory: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512
Done
  ✓ Should limit buying to swapWhitelist (44ms)
  ✓ Should activate maxBalanceAfterBuy and whitelist address (51ms)
Pair For: 0xcd905cdfdb672103d575ba3fb188ebc82cb4c7b4

```



```
factory: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512
Done
Pair For: 0xcd905cdfdb672103d575ba3fb188ebc82cb4c7b4
factory: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512
Done
Pair For: 0xcd905cdfdb672103d575ba3fb188ebc82cb4c7b4
factory: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512
Done
```

✓ Should test maxBalanceAfterBuy (181ms)

Tax Token Buy Back Tax and Settings Test

```
Tax Sigs length: 9
Lossless Sigs length: 7
Settings Sigs length: 15
Wallets Sigs length: 4
AntiBot Sigs length: 18
Multicall Sigs length: 2
Tax Sigs Duplicates: 0
Lossless Sigs Duplicates: 0
Settings Sigs Duplicates: 0
Settings Sigs Duplicates: 0
AntiBot Sigs Duplicates: 0
Multicall Sigs Duplicates: 0
All Sigs Duplicates: 0
```

Creating Pair -----

- ✓ Should create a taxToken with no settings (96ms)
- ✓ Should transfer tokens to user
- ✓ Should test basic ERC20 functions

```
0xEea07547df55BfcFcD33dF357aAE98996119D96F
```

Liquidity Time

addLiquidity

```
Pair from factory: 0xeea07547df55bfcfc33df357aae98996119d96f
```

```
factory: 0x4b6ab5f819a515382b0deb6935d793817bb4af28
```

```
Pair For: 0xeea07547df55bfcfc33df357aae98996119d96f
```

```
factory: 0x4b6ab5f819a515382b0deb6935d793817bb4af28
```

```
Done
_mintFee()
true
  ✓ Should add liquidity to the pool (44ms)
  ✓ Should turn on buyback tax in settings
  ✓ Should ignore tax on buys
  ✓ Should set a buyBack Wallet Threshold
  ✓ Should tax sells (72ms)
  ✓ Should tax sells and hit buyback threshold (113ms)
```

Custom Taxes and Settings Test

```
Tax Sigs length: 9
Lossless Sigs length: 7
Settings Sigs length: 15
Wallets Sigs length: 4
AntiBot Sigs length: 18
Multicall Sigs length: 2
Tax Sigs Duplicates: 0
Lossless Sigs Duplicates: 0
Settings Sigs Duplicates: 0
Settings Sigs Duplicates: 0
AntiBot Sigs Duplicates: 0
Multicall Sigs Duplicates: 0
All Sigs Duplicates: 0
Creating Pair -----
  ✓ Should create a taxToken (96ms)
  ✓ Should transfer tokens to user
  ✓ Should test basic ERC20 functions
0x77Ff69D03927Ec2a8580c7F269C1aCaf3B4c2DDd
Liquidity Time
addLiquidity
Pair from factory: 0x77ff69d03927ec2a8580c7f269c1acaf3b4c2ddd
factory: 0x54b8d8e2455946f2a5b8982283f2359812e815ce
Pair For: 0x77ff69d03927ec2a8580c7f269c1acaf3b4c2ddd
factory: 0x54b8d8e2455946f2a5b8982283f2359812e815ce
```

```
Done
_mintFee()
true
  ✓ Should add liquidity to the pool (44ms)
  ✓ Should fail to add custom taxes due to max fee check
  ✓ Should add custom taxes
  ✓ Should allow buys
  ✓ Should allow sells (39ms)
```

Tax Token Fees Test

```
Tax Sigs length: 9
Lossless Sigs length: 7
Settings Sigs length: 15
Wallets Sigs length: 4
AntiBot Sigs length: 18
Multicall Sigs length: 2
Tax Sigs Duplicates: 0
Lossless Sigs Duplicates: 0
Settings Sigs Duplicates: 0
Settings Sigs Duplicates: 0
AntiBot Sigs Duplicates: 0
Multicall Sigs Duplicates: 0
All Sigs Duplicates: 0
  ✓ Should set a generator fee of 1 gas token
```

```
Creating Pair -----
```

```
0x02F0DaB81A9a423308049579926C54eaC80c4018
```

```
Passed
```

```
  ✓ Should create a taxToken with no settings (95ms)
0xa0Ee7A142d267C1f36714E4a8F75612F20a79720
BigNumber { _hex: '0x02bf20', _isBigNumber: true }
  ✓ Should send the fee to the fee wallet on creation
  ✓ Should send the fee to the fee wallet on mint
  ✓ Should transfer tokens to user
  ✓ Should test basic ERC20 functions
```

```

Tax Token Holderss Test
Tax Sigs length: 9
Lossless Sigs length: 7
Settings Sigs length: 15
Wallets Sigs length: 4
AntiBot Sigs length: 18
Multicall Sigs length: 2
Tax Sigs Duplicates: 0
Lossless Sigs Duplicates: 0
Settings Sigs Duplicates: 0
Settings Sigs Duplicates: 0
AntiBot Sigs Duplicates: 0
Multicall Sigs Duplicates: 0
All Sigs Duplicates: 0
Creating Pair -----
    ✓ Should create a taxToken with no settings (104ms)
    ✓ Should check holders tax in settings
    ✓ Should transfer tokens to user
    ✓ Should test basic ERC20 functions (56ms)
0x31E51A395B6B74704Bc11b4CD9b962397708679B
Liquidity Time
addLiquidity
Pair from factory: 0x31e51a395b6b74704bc11b4cd9b962397708679b
factory: 0x398e4948e373db819606a459456176d31c3b1f91
Pair For: 0x31e51a395b6b74704bc11b4cd9b962397708679b
factory: 0x398e4948e373db819606a459456176d31c3b1f91
Done
_mintFee()
true
    ✓ Should add liquidity to the pool (53ms)
    ✓ Should exclude addresses holders tax in settings
BigNumber { _hex: '0x03e6', _isBigNumber: true }
    ✓ Should transfer tokens to user
    ✓ Should reflect on buys (51ms)
    ✓ Should reflect on buys (50ms)

```

✓ Should reflect sells (50ms)

Tax Token LP Tax and Settings Test

Tax Sigs length: 9

Lossless Sigs length: 7

Settings Sigs length: 15

Wallets Sigs length: 4

AntiBot Sigs length: 18

Multicall Sigs length: 2

Tax Sigs Duplicates: 0

Lossless Sigs Duplicates: 0

Settings Sigs Duplicates: 0

Settings Sigs Duplicates: 0

AntiBot Sigs Duplicates: 0

Multicall Sigs Duplicates: 0

All Sigs Duplicates: 0

Creating Pair -----

✓ Should create a taxToken with no settings (94ms)

✓ Should transfer tokens to user

✓ Should test basic ERC20 functions

0xFB5E117d4784dB9C76Ca599843E8531BA7b78B42

Liquidity Time

addLiquidity

Pair from factory: 0xfb5e117d4784db9c76ca599843e8531ba7b78b42

factory: 0x3abbb0d6ad848d64c8956edc9bf6f18ac22e1485

Pair For: 0xfb5e117d4784db9c76ca599843e8531ba7b78b42

factory: 0x3abbb0d6ad848d64c8956edc9bf6f18ac22e1485

Done

_mintFee()

true

✓ Should add liquidity to the pool (45ms)

✓ Should turn on lp tax in settings

✓ Should ignore tax on buys

✓ Should set a LP Wallet Threshold

✓ Should tax sells (80ms)

```
addLiquidity
Pair from factory: 0xfb5e117d4784db9c76ca599843e8531ba7b78b42
factory: 0x3abbb0d6ad848d64c8956edc9bf6f18ac22e1485
Pair For: 0xfb5e117d4784db9c76ca599843e8531ba7b78b42
factory: 0x3abbb0d6ad848d64c8956edc9bf6f18ac22e1485
Done
_mintFee()
true
k Time!!!!
sub time
  ✓ Should tax sells and hit LP threshold (134ms)
```

```
  Multicall Test
0x8b9d5A75328b5F3167b04B42AD00092E7d6c485c
Tax Sigs length: 9
Lossless Sigs length: 7
Settings Sigs length: 15
Wallets Sigs length: 4
AntiBot Sigs length: 18
Multicall Sigs length: 2
Tax Sigs Duplicates: 0
Lossless Sigs Duplicates: 0
Settings Sigs Duplicates: 0
Settings Sigs Duplicates: 0
AntiBot Sigs Duplicates: 0
Multicall Sigs Duplicates: 0
All Sigs Duplicates: 0
Creating Pair -----
  ✓ Should create a taxToken (111ms)
  ✓ Should test admin update multicall (52ms)
  ✓ Should test antiBot update multicall
```

```
  Tax Token Holders and Settings Test
Tax Sigs length: 9
Lossless Sigs length: 7
```

```
Settings Sigs length: 15
Wallets Sigs length: 4
AntiBot Sigs length: 18
Multicall Sigs length: 2
Tax Sigs Duplicates: 0
Lossless Sigs Duplicates: 0
Settings Sigs Duplicates: 0
Settings Sigs Duplicates: 0
AntiBot Sigs Duplicates: 0
Multicall Sigs Duplicates: 0
All Sigs Duplicates: 0
Creating Pair -----
  ✓ Should create a taxToken (95ms)
  ✓ Should transfer tokens to user
  ✓ Should test basic ERC20 functions
0xFFf05959C6056071f33e24470618550e49Cc4dc4
Liquidity Time
addLiquidity
Pair from factory: 0xffff05959c6056071f33e24470618550e49cc4dc4
factory: 0xfb6dab6200b8958c2655c3747708f82243d3f32e
Pair For: 0xffff05959c6056071f33e24470618550e49cc4dc4
factory: 0xfb6dab6200b8958c2655c3747708f82243d3f32e
Done
_mintFee()
true
  ✓ Should add liquidity to the pool (41ms)
  ✓ Should allow buys
  ✓ Should allow sells
  ✓ Should turn on pausing
  ✓ Should not allow buys
  ✓ Should not allow transfers
  ✓ Should turn off pausing
  ✓ Should allow transfers
  ✓ Should blacklist user 2
  ✓ Should blacklist user 2 from transfers
```

- ✓ Should allow minting
- ✓ Should turn should off pausing settings and lock settings (44ms)

Stress Test

Tax Sigs length: 9
Lossless Sigs length: 7
Settings Sigs length: 15
Wallets Sigs length: 4
AntiBot Sigs length: 18
Multicall Sigs length: 2
Tax Sigs Duplicates: 0
Lossless Sigs Duplicates: 0
Settings Sigs Duplicates: 0
Settings Sigs Duplicates: 0
AntiBot Sigs Duplicates: 0
Multicall Sigs Duplicates: 0
All Sigs Duplicates: 0

Creating Pair -----

- ✓ Should create a taxToken (119ms)
- ✓ Should set a buyBack Wallet Threshold
- ✓ Should set a LP Wallet Threshold
- ✓ Should transfer tokens to user (38ms)
- ✓ Should test basic ERC20 functions

Liquidity Time

BigNumber { value: "5000000000000000000000" }

addLiquidity

Pair from factory: 0x17a5bcbfca3aed0f0bcfa825fc4b9677392abef3

factory: 0x1966dc8ff30bc4aeded27178642253b3ccc9aa3f

Pair For: 0x17a5bcbfca3aed0f0bcfa825fc4b9677392abef3

factory: 0x1966dc8ff30bc4aeded27178642253b3ccc9aa3f

Done

_mintFee()

true

- ✓ Should add liquidity to the pool (110ms)

0x17a5BcbFCa3AEd0f0BCfa825Fc4B9677392ABef3


```

0x5f58879Fe3a4330B6D85c1015971Ea6e5175AeDD
0.0000000000000001
47.910455016924957361
47.910455016924958361
Approval
0xf39Fd6e51aad88F6F4ce6aB8827279cFfFb92266
0x5f58879Fe3a4330B6D85c1015971Ea6e5175AeDD
removeLiquidityETHSupportingFeeOnTransferTokens
removeLiquidity
Pair: 0x17a5bcfbca3aed0f0bcfa825fc4b9677392abef3
sender: 0xf39fd6e51aad88f6f4ce6ab8827279cfff92266
liquidity: 1000000000000000000
allowance[from][msg.sender] : 1000000000000000000
value : 1000000000000000000
liquidity sent to pair
buuuuurn
_mintFee()
true
k Time!!!!
[
  BigNumber { _hex: '0x185e11da5f152d415f', _isBigNumber: true },
  BigNumber { _hex: '0x43f0cd68d52a6af1', _isBigNumber: true },
  1673114094,
  _reserve0: BigNumber { _hex: '0x185e11da5f152d415f', _isBigNumber: true },
  _reserve1: BigNumber { _hex: '0x43f0cd68d52a6af1', _isBigNumber: true },
  _blockTimestampLast: 1673114094
]
BigNumber {
  _hex: '0x06778a6a32691c04a1e13db4b07141e06f',
  _isBigNumber: true
}
  ✓ Should remove liquidity from the pool (139ms)

```

Tax Token No Settings Test
Tax Sigs length: 9

```
Lossless Sigs length: 7
Settings Sigs length: 15
Wallets Sigs length: 4
AntiBot Sigs length: 18
Multicall Sigs length: 2
Tax Sigs Duplicates: 0
Lossless Sigs Duplicates: 0
Settings Sigs Duplicates: 0
Settings Sigs Duplicates: 0
AntiBot Sigs Duplicates: 0
Multicall Sigs Duplicates: 0
All Sigs Duplicates: 0
Creating Pair -----
0xbf558FB5fCF987fcDbC72938c790aE3d140f4Fe6
Passed
  ✓ Should create a taxToken with no settings (92ms)
  ✓ Should send the fee to the fee wallet on mint
  ✓ Should transfer tokens to user
  ✓ Should test basic ERC20 functions
```

Tax Token Transaction Tax Test

```
Tax Sigs length: 9
Lossless Sigs length: 7
Settings Sigs length: 15
Wallets Sigs length: 4
AntiBot Sigs length: 18
Multicall Sigs length: 2
Tax Sigs Duplicates: 0
Lossless Sigs Duplicates: 0
Settings Sigs Duplicates: 0
Settings Sigs Duplicates: 0
AntiBot Sigs Duplicates: 0
Multicall Sigs Duplicates: 0
All Sigs Duplicates: 0
Creating Pair -----
```

```
✓ Should create a taxToken with no settings (334ms)
✓ Should transfer tokens to user
✓ Should test basic ERC20 functions
0xF5AC4122828b9C5D1fda7a8d535651cc904DF7c5
Liquidity Time
addLiquidity
Pair from factory: 0xf5ac4122828b9c5d1fda7a8d535651cc904df7c5
factory: 0x81a5186946ce055a5ceec93cd97c7e7ede7da922
Pair For: 0xf5ac4122828b9c5d1fda7a8d535651cc904df7c5
factory: 0x81a5186946ce055a5ceec93cd97c7e7ede7da922
Done
_mintFee()
true
✓ Should add liquidity to the pool (47ms)
✓ Should tax buys
✓ Should tax sells (40ms)
✓ Should confirm funds in tax wallet
```

90 passing (16s)

7. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase.

The main goal of the audit was to verify the claims regarding the security and functions of the smart contract. During the audit, no critical, no high, four medium, seven low and eight informational issues have been found, after the manual and automated security testing.

We advise the Unicrypt team to implement the recommendations to further enhance the code's security and readability.

Update (11.01.2023): Unicrypt team fixed all necessary issues.

8. About the Auditor

Chainsulting is a professional software development firm, founded in 2017 and based in Germany. They show ways, opportunities, risks and offer comprehensive Web3 solutions. Their services include web3 development, security and consulting.

Chainsulting conducts code audits on market-leading blockchains such as Solana, Tezos, Ethereum, Binance Smart Chain, and Polygon to mitigate risk and instil trust and transparency into the vibrant crypto community. They have also reviewed and secure the smart contracts of many 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 web3 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>

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.