expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

Lenny Token

Audited On : 11 Jan, 2024

EXECUTIVE SUMMARY

Risk Findings

Severity	Found
High	1
Medium	1
Low	3
Informational	1

Owner Privileges

Set Taxes and Ratios, Airdrop, Set Protection Settings, Set Reward Properties, Set Reflector Settings, Set Swap Settings, Set Pair and Router

---

LENNYTOKEN Smart contract has achieved the following score:

95.0

- Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.
- Please note that centralization privileges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.

TABLE OF CONTENTS

SCOPE OF WORK

Expelee was consulted by LENNY TOKEN to conduct the smart contract audit of it's Rust source code.

The audit scope of work is strictly limited to mentioned .Rust file only:

In the following, we show the specific pull request and the commit hash value used in this audit.

- https://github.com/vinhtranz/cremation-coin/tree/main/contracts/lenny_token
- https://github.com/vinhtranz/cremation-coin/blob/main/contracts/lenny_token/Cargo.toml

External contracts and/or interfaces dependencies are not checked due to being out of scope.

Public Contract Links

https://github.com/vinhtranz/cremation-coin/blob/main/contracts/lenny_token/Cargo.toml

https://github.com/vinhtranz/cremation-coin/blob/main/contracts/lenny_token/src/testing.rs

https://github.com/vinhtranz/cremation-coin/blob/main/contracts/lenny_token/src/state.rs

https://github.com/vinhtranz/cremation-coin/blob/main/contracts/lenny_token/src/msg.rs

https://github.com/vinhtranz/cremation-coin/blob/main/contracts/lenny_token/src/lib.rs

https://github.com/vinhtranz/cremation-coin/blob/main/contracts/lenny_token/src/contract.rs

https://github.com/vinhtranz/cremation-coin/blob/main/contracts/lenny_token/src/bin/schema.rs

AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effectivesmart contract audit. Here's a brief overview of Expelee auditing process and methodology:

CONNECT

The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.

AUDIT

- Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:
    - Remix IDE DeveloperTool
    - Open Zeppelin Code Analyzer
    - SWC Vulnerabilities Registry
    - DEX Dependencies, e.g., Pancakeswap,Uniswap
- Simulations are performed to identify centralized exploits causing contract and/or trade locks.
- A manual line-by-line analysis is performed to identify contract issues and centralized privileges. We may inspect below mentioned common contract vulnerabilities, and centralized exploits:

Centralized Exploits

- Token Supply Manipulation
- Access Control and Authorization
- Assets Manipulation
- Ownership Control
- Liquidity Access
- Stop and PauseTrading
- Ownable Library Verification

Common Contract Vulnerabilities

- Integer Overflow
- Lack of Arbitrary limits
- Incorrect Inheritance Order
- Typographical Errors
- Requirement Violation
- Gas Optimization
- Coding StyleViolations
- Re-entrancy
- Third-Party Dependencies
- Potential Sandwich Attacks
- Irrelevant Codes
- Divide before multiply
- Conformance to Solidity Naming Guides
- Compiler Specific Warnings
- Language Specific Warnings

REPORT

- **It is important to note that there is no pass or fail in the audit, it is recommended to view the audit as an unbiased assessment of the safety of rust codes**

- **The client may use the audit report internally or disclose it publicly.**

expelee

FULL AUDIT CHECKLIST

Category	Checklist Items
Basic Coding Bugs	Constructor Mismatch
	Ownership Takeover
	Redundant Fallback Function
	Overflows & Underflows
	Reentrancy
	Money-Giving Bug
	Blackhole
	Unauthorized Self-Destruct
	Revert DoS
	Unchecked External Call
	Gasless Send
	Send Instead Of Transfer
	Costly Loop
	(Unsafe) Use Of Untrusted Libraries
	(Unsafe) Use Of Predictable Variables
	Transaction Ordering Dependence
	Deprecated Uses
Semantic Consistency Checks	Semantic Consistency Checks
Advanced DeFi Scrutiny	Business Logics Review
	Functionality Checks
	Authentication Management
	Access Control & Authorization
	Oracle Security
	Digital Asset Escrow
	Kill-Switch Mechanism
	Operation Trails & Event Generation
	ERC20 Idiosyncrasies Handling
	Frontend-Contract Integration
	Deployment Consistency
	Holistic Risk Management
Additional Recommendations	Avoiding Use of Variadic Byte Array
	Using Fixed Compiler Version
	Making Visibility Level Explicit
	Making Type Inference Explicit
	Adhering To Function Declaration Strictly
	Following Other Best Practices

VULNERABILITY CHECKS

Compiler errors	Passed
Race Conditions and reentrancy. Cross-Function Race Conditions	Passed
Possible Delay In Data Delivery	Passed
Oracle calls	Passed
Front Running	Passed
Sol Dependency	Passed
Integer Overflow And Underflow	Passed
DoS with Revert	Passed
Dos With Block Gas Limit	Passed
Methods execution permissions	Passed
Economy Model of the contract	Passed
The Impact Of Exchange Rate On the solidity Logic	Passed
Private use data leaks	Passed
Malicious Event log	Passed
Design Logic	Passed
Scoping and declarations	Passed
Uninitialized storage pointers	Passed
Arithmetic accuracy	Passed
Cross-function race conditions	Passed
Save Upon solidity contract Implementation and Usage	Passed

RISK CLASSIFICATION

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and acces control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time. We categorize these vulnerabilities by the following levels:

High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Low Risk

Issues on this level are minor details and warning that can remain unfixed.

Informational

Issues on this level are minor details and warning that can remain unfixed.

Status Type	Definition
Open	Risks are open.
Acknowledged	Risks are acknowledged, but not fixed.
Resolved	Risks are acknowledged and fixed.

expelee

AUDIT SCOPE

ID	Repo	Comment	File	SHM321 Checksum
LBY	contracts/lenny_token/src /contract.rs	cC512486	Contract.rs	6788099YIRHVSK853PKFMGHEF443092 00KDHFCBUGIJN
LBI	contracts/lenny_token/src /contract.rs	cC512486	Contract.rs	347520JHDB7549H22H3BVDIOETYUHF 009JBIKBDI33BJ4
LBW	contracts/lenny_token/src /contract.rs	cC512486	Contract.rs	1988Y73HUGFDINN353840NFMTEJER7 3649RGFIMDIDH
LBG	contracts/lenny_token/src /contract.rs	cC512486	Contract.rs	4438648TEOHBF6378309EHROECNEPO EJDNETE8EYEU3
LBL	contracts/lenny_token/src /contract.rs	cC512486	Contract.rs	66390028765RVNKDBYFTGW553T2KO EHIUUJJIJE
LBA	contracts/lenny_token/src /contract.rs	cC512486	Contract.rs	09825539BDYG543DVNKOMIKEBYR JUFHHFHJFIE333222
LBJ	contracts/lenny_token/src /contract.rs	cC512486	Contract.rs	8654RJVT3DWI865YK26437903JJDGGD HGWY6E
LBE	contracts/lenny_token/src /testing.rs	cC512486	testing.rs	7763888636TGYGFFTFHBETT66TFTCTV YBHBYT
LBP	contracts/lenny_token/src /testing.rs	cC512486	testing.rs	88530486494YRHFTEICBGEIEGWTWY WUHEJEHEIE33U3
LBM	contracts/lenny_token/src /testing.rs	cC512486	testing.rs	1209873KHJLKJNFJHGE9876399002977 4BCUHHDUU239
LBV	contracts/lenny_token/src /testing.rs	cC512486	testing.rs	23456UGFYUHE98756EFHJHE7654ESDF GHGERTYUJ3897
LBQ	contracts/lenny_token/src /state.rs	cC512486	state.rs	37889UHBIONEO7TYRDFGVBN5678939 IJWSFVDYUHDCI
LBS	contracts/lenny_token/src /state.rs	cC512486	state.rs	678903098TFHJKFCPOIUGFGHJKE9865 ERGBEIVBHE8767
LBR	contracts/lenny_token/src /state.rs	cC512480	state.rs	98765SDFGBNFCOI56789UIYHGGHEJDI UYTRDCVBN3459
LCD	contracts/lenny_token/src /lib.rs	cC512481	lib.rs	3348y9808hgtrusvnmu43100ejfojg fnut8496230hb574he
LHU	contracts/lenny_token/src /lib.rs	cC512481	lib.rs	9864byf5f379eig28ffre64085jv161 3251guhkdmue87
LGG	contracts/lenny_token/src /msg.rs	cC512481	msg.rs	7ej2d8jg765tjfiowg538ij74dwftyv6 478ij3gs820
LTR	contracts/lenny_token/src /msg.rs	cC512481	msg.rs	864fr46de438hdguw903rfdcb246d buhb2917enk

AUTOMATED ANALYSIS

Symbol	Definition
	Function modifies state
	Function is payable
	Function is internal
	Function is private
	Function is important

LENNY TOKEN	Interface			
└	totalSupply	External		!	NO	
└	decimals	External		!	NO	
└	symbol	External		!	NO	
└	name	External		!	NO	
└	getOwner	External			NO	
└	balanceOf	External		!	NO	
└	transfer	External	"	!		NO	
└	allowance	External		!	NO	
└	approve	External	"	!		NO	
└	transferFrom	External	"		NO	

IFactoryV2	Interface				
└	getPair	External		NO	
└	createPair	External	"	NO	

IV2Pair	Interface				
└	factory	External		NO	
└	getReserves	External		NO	
└	sync	External	"	NO	

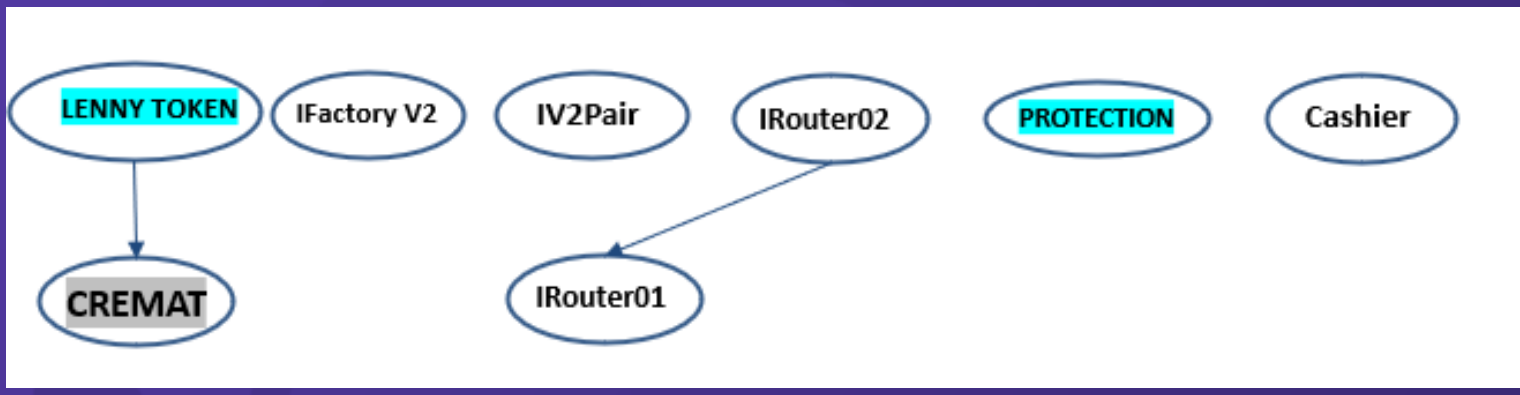
AUTOMATED ANALYSIS

IRouter01	Interface		
└	factory	External	NO
└	dex	External	NO
└	addLiquiditydex	External	# NO
└	addLiquidity	External	" NO
└	swapExactdexForTokens	External	# NO
└	getAmountsOut	External	NO
└	getAmountsIn	External	NO
IRouter02	Interface	IRouter01	
└	swapExactTokensFordexSupportingFeeOnTransferTokens	External	" NO
└	swapExactdexForTokensSupportingFeeOnTransferTokens	External	# NO
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	" ! NO
└	swapExactTokensForTokens	External	" NO
Protections	Interface		
└	checkUser	External	" ! NO
└	setLaunch	External	" ! NO
└	setLpPair	External	" ! NO
└	CREMAT	External	" NO
└	removeSniper	External	" NO
Cashier	Interface		
└	setRewardsProperties	External	" NO
└	tally	External	" NO
└	load	External	# NO
└	cashout	External	" NO
└	giveMeWelfarePlease	External	" NO
└	getTotalDistributed	External	NO
└	getUserInfo	External	NO
└	getUserRealizedRewards	External	NO

AUTOMATED ANALYSIS

└	getPendingRewards	External ❗		|NO❗
└	initialize	External ❗	"	|NO❗
└	getCurrentReward	External ❗		|NO❗
rs	Implementation	**SafeMath**		
└	<Constructor>	Public ❗		# |NO❗
└	transferOwner	External ❗	"		onlyOwner
└	renounceOwnership	External ❗	"		NO❗
└	setOperator	Public ❗	"	|NO❗
└	renounceOriginalDeployer	External ❗	"	|NO❗
└	<Receive Ether>	External ❗		# |NO❗
└	totalSupply	External ❗		|NO❗
└	decimals	External ❗		|NO❗
└	symbol	External ❗		|NO❗
└	name	External ❗		|NO❗
└	getOwner	External ❗		! |NO❗
└	balanceOf	Public ❗		! |NO❗
└	allowance	External ❗		! |NO❗
└	approve	External ❗	"	! 🔴 |NO❗
└	_approve	Internal $	"		
└	approveContractContingency	Public ❗	"	! 🔴	onlyOwner
└	transfer	External ❗	"	|NO❗
└	transferFrom	External ❗	"	|NO❗
└	setNewRouter	External ❗	"		onlyOwner
└	setLpPair	External ❗	"		onlyOwner
└	setInitializers	External ❗	"		onlyOwner
└	isExcludedFromFees	External ❗		|NO❗
└	isExcludedFromDividends	External ❗		|NO❗
└	isExcludedFromProtection	External ❗		|NO❗
└	setDividendExcluded	Public ❗	"		onlyOwner
└	setExcludedFromFees	Public ❗	"		onlyOwner

INHERITANCE GRAPH



Vulnerability 0 : No important security issue detected.
Threat level: Low

.gitignore | testing.rs | contract.rs

```
280     }
281
282     pub fn set_tax_free_address(
283         deps: DepsMut,
284         _env: Env,
285         info: MessageInfo,
286         address: String,
287         tax_free: bool,
288     ) -> Result<Response, ContractError> {
289         let owner = OWNER.load(deps.storage)?;
290         if info.sender != owner {
291             return Err(ContractError::Unauthorized {});
292         }
293         let address = deps.api.addr_validate(&address)?;
294         TAX_FREE_ADDRESSES.save(deps.storage, address, &tax_free)?;
295         Ok(Response::new())
296     }
297
298     pub fn send(
```

OPTIMIZATIONS

ID	Title	Category	Status
STV	Logarithm Refinement Optimization	Gas Optimization	Acknowledged
SOP	Checks Can Be Performed Earlier	Gas Optimization	Acknowledged
SDP	Unnecessary Use Of SafeMath	Gas Optimization	Acknowledged
SWY	Struct Optimization	Gas Optimization	Acknowledged
SGT	Unused State Variable	Gas Optimization	Acknowledged

MANUAL REVIEW

Severity Criteria

Expelee assesses the severity of disclosed vulnerabilities according to methodology based on OWASP standarts.

Vulnerabilities are dividend into three primary risk categroies:
High
Medium
Low

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious input handling
- Escalation of privileges
- Arithmetic
- Gas use

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

LNY-01 KEY FINDINGS

Category: Status Mathematical Operations
Severity: Low
Target: Multiple Contracts
Status: Informational

Description

In **updateForMinter** , the following equation is used inside an unchecked block

```
};
  let token_info_res: TokenInfoResponse =
        deps.querier.query(&QueryRequest::Wasm(token_info_query))?;
 let maximum_supply = minter_res.cap.unwrap();
 let current_supply = token_info_res.total_supply;
 let mintable_amount = maximum_supply - current_supply;
```

Minter can not issue more **Lenny Tokens** indefinitely.
Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the **LENNY TOKEN** contract.

Thus, this enables the approval of a token account for confidential transfers, even if it is associated with a different mint. Ideally, token accounts should only be allowed to hold tokens from the specific mint they are associated with. By not checking the mint consistency, the function effectively approves arbitrary token accounts for confidential transfers. Such unauthorized token mixing may have security and financial implications, as it could result in loss of value or assets for users who rely on the token system's integrity.

Recommendation

Incorporate the following verification within process_approve_account to confirm that the token account's associated mint aligns with the mint for which the confidential transfer approval is sought.

LNY-02 KEY FINDINGS

Category: Business Logic
Severity: Medium
Target: Contract.rs
Status: Fixed

Description

In **updateForTokenTax**, Relevant Function Snippet

```
} => execute::update_collecting_tax_address(deps, env, info, new_collect_tax_addr),
        ExecuteMsg::UpdateTaxInfo {
                buy_tax,
                sell_tax,
                transfer_tax,
    } => execute::update_tax_info(deps, env, info, buy_tax, sell_tax, transfer_tax),
    ExecuteMsg::SetTaxFreeAddress { address, tax_free } => {
    execute::set_tax_free_address(deps, env, info, address, tax_free)
```

Description

Tax() should be declared external: -
totalSupply() should be declared external:
- LENNY TOKEN.totalSupply() (Contract.rs#67-74

Recommendation

We recommend either checking for overflow in this case, or ensuring that the PairsIn is close enough it will never cause an overflow

LNY-03 KEY FINDINGS

Category: Inconsistency
Severity: Informational
Target: Multiple Contracts
Status: Acknowledge

Description

In **updateForMinter**, Relevant Function Snippet

```
let res: Response = Response::new()
        .add_message(WasmMsg::Execute {
                contract_addr: token_address.clone().into(),
                msg: to_binary(&withdraw_cw20_msg)?,
                funds: vec![],
        })
        .add_attribute("action", "withdraw")
        .add_attribute("token_address", token_address)
        .add_attribute("amount", locked_amount);
  Ok(res)
```

Description

The function amount0 () does not have the override specifier. It should be noted that since amount0 > a function That overrides only a single interface function does not require the override specifier. However, all other instances of thisin the codebase contain the override specifier

Recommendation

We recommend adding the override specifier to amount() or removing the override specifier from all other functions this applies to for consistency.

LNY-04 KEY FINDINGS

Category: Coding Practices
Severity: High
Target: contracts/lenny_token/src/contract.rs
Status: Acknowledge

Description

In **updateForAsset**, Relevant Function Snippet

```
contract: router.to_string(),
        amount: collected_tax_amount,
        msg: to_binary(&RouterExecuteMsg::ExecuteSwapOperations {
          operations: vec![
              SwapOperation::TerraSwap {
                offer_asset_info: AssetInfo::Token {
                        contract_addr: env.contract.address.to_string(),
                },
                ask_asset_info: AssetInfo::NativeToken {
                        denom: "uluna".to_string(),
```

Description

For any Asset Trading Platform, there is a need to reliably and accurately measure the Asset trading debt position and provide necessary means to liquidate underwater positions. The Lenny Token platform is no exception. While reviewing the implementation to measure the debt position, we notice the key function offer_asset_info: AssetInfo::Token () needs to be improved.

Recommendation

Apply the right price oracle in the above offer_asset_info: AssetInfo::Token () routine to compute the user account data

LNY-05 KEY FINDINGS

Category: Coding Practices
Severity: Low
Target: contracts/lenny_token/src/testing.rs
Status: Confirmed

Description

updateForbalance, Relevant Function Snippet

```
    let sender_balance_after = helpers::query_balance(&deps, &sender);
                let recipient_balance_after = helpers::query_balance(&deps,
 &recipient);
                let collect_tax_wallet_balance_after =
                    helpers::query_balance(&deps, &collect_tax_wallet);
```

Description

While re-viewing arithmetic operations in current balance implementation, we notice occasion that may introduce unexpected overflows/underflows. For example, if we examine the helpers::query_balance(&deps, &sender); function, it may revert if the current collateralData (line 821) is equal to 0. Another example is when the underlying asset of a recipient has an unusual decimal, which may revert the following calculation of if sender == &collect_tax_wallet || recipient == &collect_tax_wallet {
                    assert_eq!(

Note this calculation appears in a num- ber of routines. Its revert may bring in unnecessary frictions and cause issues for integration and composability.

Recommendation

Revise the above calculation to avoid the unnecessary overflows and under- flows.

LNY-06 KEY FINDINGS

Category: Coding Practices
Severity: Low
Target: contracts/lenny_token/src/contract.rs
Status: Informational

Description

In **updateForOwner**, Relevant Function Snippet

```
 let new_owner = deps.api.addr_validate(&new_owner)?;
    OWNER.save(deps.storage, &new_owner)?;
    Ok(Response::new())
 }

 pub fn update_collecting_tax_address(
    deps: DepsMut,
    _env: Env,
    info: MessageInfo,
    new_collect_tax_addr: String,
 ) -> Result<Response, ContractError> {
    let owner = OWNER.load(deps.storage)?;
    if info.sender != owner {
        return Err(ContractError::Unauthorized {});
    }
```

Description

For Ownership efficiency, the Lenny Token is engineered with the reserve cache mechanism, which necessi-tates the common steps to be followed when operating with the reserve Ownership data in different scenarios, including the tax generation, update, and eventual persistence.

Recommendation

Revise the above functions to following a consistent approach to use the reserve cache mechanism.

ABOUT EXPELEE

Expelee is a product-based aspirational Web3 start-up. Coping up with numerous solutions for blockchain security and constructing a Web3 ecosystem from deal making platform to developer hosting open platform, while also developing our own commercial and sustainable blockchain.

www.expelee.com

expeleeofficial

expelee

Expelee

expelee

expelee_official

expelee-co

expelee

Building the Futuristic Blockchain Ecosystem

DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantess against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always do your own research and project yourselves from being scammed. The Expelee team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools.

Under no circumstances did Expelee receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Alway do your own research and protect yourselves from scams.

This document should not be presented as a reason to buy or not buy any particular token. The Expelee team disclaims any liability for the resulting losses.