CFG NINJA AUDITS

Security Assessment

KTULU Token

January 10, 2023

Table of Contents

Assessment Summary

This report has been prepared for KTULU Token on the Ethereum network. CFGNINJA provides both client-centered and user-centered examination of the smart contracts and their current status when applicable. This report represents the security assessment made to find issues and vulnerabilities on the source code along with the current liquidity and token holder statistics of the protocol.

A comprehensive examination has been performed, utilizing Cross Referencing, Static Analysis, In-House Security Tools, and line-by-line Manual Review.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Inspecting liquidity and holders statistics to inform the current status to both users and client when applicable.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Verifying contract functions that allow trusted and/or untrusted actors to mint, lock, pause, and transfer assets.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders
- Thorough line-by-line manual review of the entire codebase by industry experts.

Technical Findings Summary

Classification of Risk

Severity	Description
Critical	Risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
Major	Risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
Medium	Risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
Minor	Risks can be any of the above but on a smaller scale. They generally do not compromise the overall integrity of the Project, but they may be less efficient than other solutions.
Informational	Errors are often recommended to improve the code's style or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

Findings

Severity	Found	Pending	Resolved
Critical	0	0	0
Major	0	0	0
Medium	0	0	0
Minor	0	0	0
Informational	0	0	0
Total	0	0	0

Project Overview

Token Summary

Parameter	Result
Address	0xaa3ac37D7d29fB54CE464E612b02c0598E04298A
Name	KTULU
Token Tracker	KTULU ($KTL)
Decimals	18
Supply	1,000,000,000,000
Platform	Ethereum
compiler	v0.8.15+commit.e14f2714
Contract Name	Ktulu
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://etherscan.io/address/0xaa3ac37D7d29fB54CE464E612b02c0598E04298A#code
Payment Tx	0xd65277598871f655086e89862fd5aa31bef8aa91dcdb076eb01822db9b67f84d

Project Overview

Risk Analysis Summary

Parameter	Result
Buy Tax	2%
Sale Tax	2%
Is honeypot?	Clean
Can edit tax?	No
Is anti whale?	Yes
Is blacklisted?	No
Is whitelisted?	Yes
Holders	170
Confidence Level	Medium

The following quick summary it's added to the project overview; however, there are more details about the audit and its results. Please read every detail.

Main Contract Assessed
Contract Name

Name	Contract	Live
KTULU	0xaa3ac37D7d29fB54CE464E612b02c0598E04298A	Yes

TestNet Contract Assessed
Contract Name

Name	Contract	Live
KTULU	0xfaeab522C2095c70A8CE5D220D80F9Abd16431D4	Yes

Solidity Code Provided

SolID	File Sha-1	FileName
Ktulu	83e60c86224cde77cfb0b77e44434cc59e6ed5dd	Ktulu.sol

Mint Check

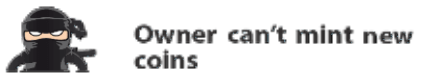
The project owners of KTULU do not have a mint function in the contract, owner cannot mint tokens after initial deploy.

The Project has a Total Supply of 1,000,000,000,000 and cannot mint any more than the Max Supply.

Mint Notes:

Auditor Notes:

Project Owner Notes:



Fees Check

The project owners of KTULU have the ability to set higher than 25%
We Recommend the team to create a new contract with fees restrictions to avoid any problems, as alternative the team can use multi signature wallet to ensure the project is safe from a potential fee increase.

Tax Fee Notes:

Auditor Notes: The contract currently has 5% buy and 12% sale taxes, and cannot be set higher than 25%.

Project Owner Notes:

Blacklist Check

The project owners of KTULU do not have a blacklist function their contract.

The Project allow owners to transfer their tokens without any restrictions.

Token owner cannot blacklist the contract: Malicious or compromised owners can trap contracts relying on tokens with a blacklist.

Blacklist Notes:

Auditor Notes: blacklist is set for those tx origin from early buy

Project Owner Notes: undefined

MaxTx Check

The Project Owners of KTULU can set max tx amount.

The ability to set MaxTx can be used as bad actor, this can limit the ability of investors to sale their tokens at any given time if is set too low..

We recommend the project to set MaxTx to Total Supply or simiar to avoid swap or transfer from failures

MaxTX Notes:

Auditor Notes:

Project Owner Notes:

Pause Trade Check

The Project Owners of KTULU can stop or pause trading

We recommend the Team only allow Open Trade and never use Stop Trade, as this will be catastrophic for the Project and Investors.

We recommend the Team create a new contract without the stop trade function.

Pause Trade Notes:

Auditor Notes: There is an Open Trade so holders cant trade until is enable.

Project Owner Notes:

Owner can pause trading

Contract Ownership

The contract ownership of KTULU has been renounced.

Having no owner means that all the ownable functions in the contract can not be called by anyone, this often leads to more trust on the project.

Liquidity Ownership

The token does not have liquidity at the moment of the audit, block 23906970

If liquidity is unlocked, then the token developers can do what is infamously known as 'rugpull'. Once investors start buying token from the exchange, the liquidity pool will accumulate more and more coins of established value (e.g., ETH or BNB or Tether). This is because investors are basically sending these tokens of value to the exchange, to get the new token. Developers can withdraw this liquidity from the exchange, cash in all the value and run off with it. Liquidity is locked by renouncing the ownership of liquidity pool (LP) tokens for a fixed time period, by sending them to a time-lock smart contract. Without ownership of LP tokens, developers cannot get liquidity pool funds back. This provides confidence to the investors that the token developers will not run away with the liquidity money. It is now a standard practice that all token developers follow, and this is what really differentiates a scam coin from a real one.

Read More

KYC Information

The Project Owners of KTULU is not KYC.

KYC Information Notes:

Auditor Notes:

Project Owner Notes:

Smart Contract Vulnerability Checks

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	Ktulu.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	Ktulu.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	Ktulu.sol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	Ktulu.sol	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	Ktulu.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	Ktulu.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	Ktulu.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	Ktulu.sol	L: 0 C: 0
SWC-108	Low	State variable visibility is not set..	Ktulu.sol	L: 266 C: 12, L: 267 C: 12
SWC-109	Pass	Uninitialized Storage Pointer.	Ktulu.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	Ktulu.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	Ktulu.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	Ktulu.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-113	Pass	Multiple calls are executed in the same transaction.	Ktulu.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	Ktulu.sol	L: 0 C: 0
SWC-115	Low	Authorization through tx.origin.	Ktulu.sol	L: 530 C: 83, L: 531 C: 75
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	Ktulu.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	Ktulu.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	Ktulu.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	Ktulu.sol	L: 0 C: 0
SWC-120	Low	Potential use of block.number as source of randonmness.	Ktulu.sol	L: 383 C: 7, L: 530 C: 96, L: 530 C: 151, L: 532 C: 7, L: 533 C: 7, L: 617 C: 37
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	Ktulu.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	Ktulu.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	Ktulu.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	Ktulu.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-125	Pass	Incorrect Inheritance Order.	Ktulu.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	Ktulu.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	Ktulu.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	Ktulu.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	Ktulu.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	Ktulu.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	Ktulu.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	Ktulu.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	Ktulu.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	Ktulu.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	Ktulu.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	Ktulu.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.

Smart Contract Vulnerability Details

SWC-108 - State Variable Default Visibility

CWE-710: Improper Adherence to Coding Standards

Description:

Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

Remediation:

Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

References:

Ethereum Smart Contract Best Practices - Explicitly mark visibility in functions and state variables

Smart Contract Vulnerability Details

SWC-115 - Authorization through tx.origin

CWE-477: Use of Obsolete Function

Description:

tx.origin is a global variable in Solidity which returns the address of the account that sent the transaction. Using the variable for authorization could make a contract vulnerable if an authorized account calls into a malicious contract. A call could be made to the vulnerable contract that passes the authorization check since tx.origin returns the original sender of the transaction which in this case is the authorized account.

Remediation:

tx.origin should not be used for authorization. Use msg.sender instead.

References:

Solidity Documentation - tx.origin

Ethereum Smart Contract Best Practices - Avoid using tx.origin

SigmaPrime - Visibility.

Smart Contract Vulnerability Details

SWC-120 - Weak Sources of Randomness from Chain Attributes

CWE-330: Use of Insufficiently Random Values

Description:

Solidity allows for ambiguous naming of state variables when inheritance is used. Contract A with a variable x could inherit contract B that also has a state variable x defined. This would result in two separate versions of x, one of them being accessed from contract A and the other one from contract B. In more complex contract systems this condition could go unnoticed and subsequently lead to security issues.

Shadowing state variables can also occur within a single contract when there are multiple definitions on the contract and function level.

Remediation:

Using commitment scheme, e.g. RANDAO. Using external sources of randomness via oracles, e.g. Oraclize. Note that this approach requires trusting in oracle, thus it may be reasonable to use multiple oracles. Using Bitcoin block hashes, as they are more expensive to mine.

References:

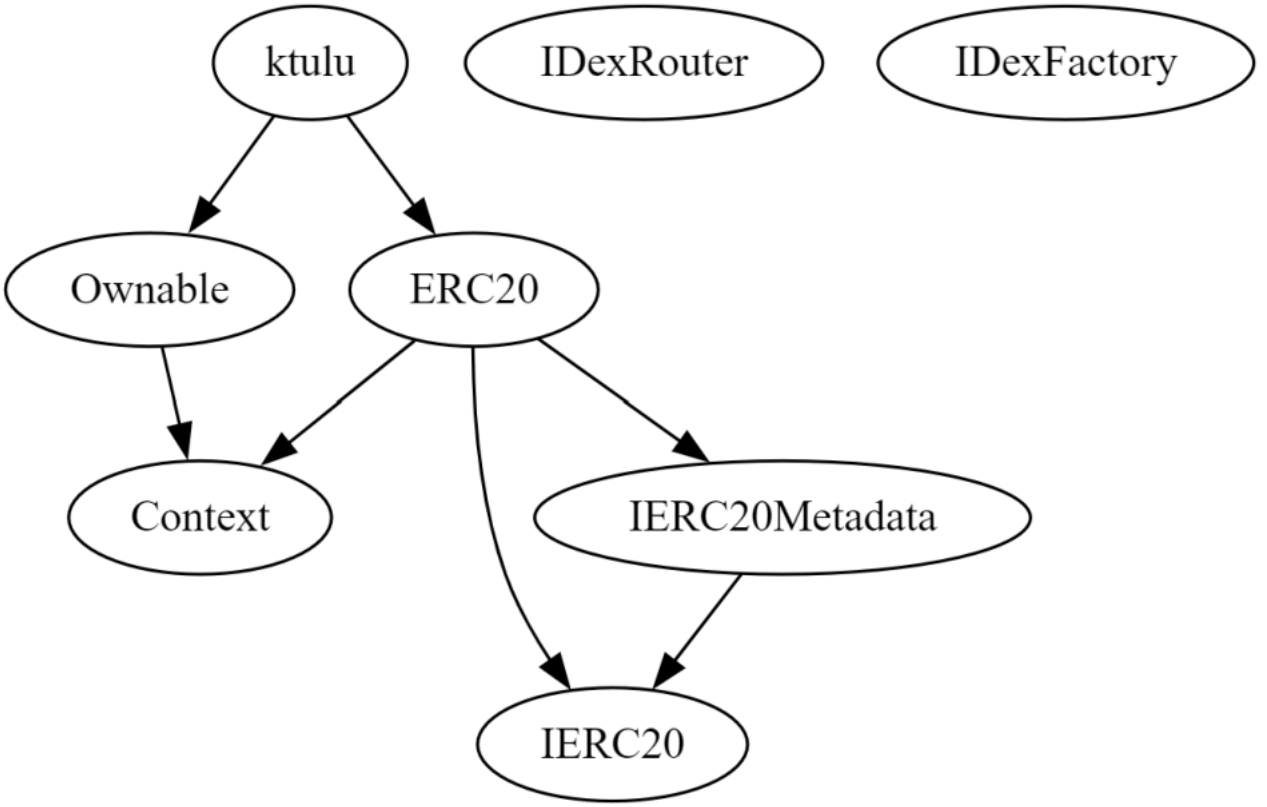
How can I securely generate a random number in my smart contract?)

When can BLOCKHASH be safely used for a random number? When would it be unsafe?

The Run smart contract.

Inheritance

The contract for KTULU has the following inheritance structure.



Smart Contract Advance Checks

ID	Severity	Name	Result	Status
$KTL-01	Minor	Potential Sandwich Attacks.	Fail	Pending
$KTL-02	Minor	Function Visibility Optimization	Pass	Pending
$KTL-03	Minor	Lack of Input Validation.	Pass	Resolved
$KTL-04	Major	Centralized Risk In addLiquidity.	Pass	Pending
$KTL-05	Major	Missing Event Emission.	Pass	Pending
$KTL-06	Minor	Conformance with Solidity Naming Conventions.	Pass	Not Found
$KTL-07	Minor	State Variables could be Declared Constant.	Pass	In-Review
$KTL-08	Major	Dead Code Elimination.	Pass	Not-Found
$KTL-09	Major	Third Party Dependencies.	Pass	Not Found
$KTL-10	Major	Initial Token Distribution.	Pass	Pending
$KTL-11	Critical	distributeTokensBetween Holders is a multisender of tokens from contract.	Pass	Resolved
$KTL-12	Major	Centralization Risks In The X Role	Pass	Not Found
$KTL-13	Informational	Extra Gas Cost For User..	Pass	Pending
$KTL-14	Informational	Unnecessary Use Of SafeMath	Pass	Pending

$KTL-01 | Potential Sandwich Attacks.

Category	Severity	Location	Status
Security	Minor	Ktulu.sol: 742,13	Pending

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by back running (after the transaction being attacked) a transaction to sell the asset. The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- swapExactTokensForETHSupportingFeeOnTransferTokens()
- addLiquidityETH()

Remediation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

Referrences:

What Are Sandwich Attacks in DeFi — and How Can You Avoid Them?.

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/TitanKtulu	Pass
Other	https://discord.gg/mZ8HPE7xVc, https://medium.com/@titan_68262, https://www.youtube.com/@ktuluerc20	Pass
Website	https://ktu-lu.com/	Pass
Telegram	https://t.me/ktuluerc20	Pass

We recommend to have 3 or more social media sources including a completed working websites.

Social Media Information Notes:

Auditor Notes: undefined

Project Owner Notes:

Assessment Results

Score Results

Review	Score
Overall Score	82/100
Auditor Score	85/100
Review by Section	**Score**
Manual Scan Score	22/35
SWC Scan Score	34 /37
Advance Check Score	26 /28

The Following Score System Has been Added to this page to help understand the value of the audit, the maximun score is 100, however to attain that value the project most pass and provide all the data needed for the assessment. Our Passing Score has been changed to 80 Points, if a project does not attain 80% is an automatic failure. Read our notes and final assessment below.

Audit Passed

Assessment Results

Important Notes:

- Contract has taxes up to 2%.
- Owner can't set max tx amount.
- No high-risk Exploits/Vulnerabilities Were Found in the Source Code.
- Owner has provided the link to the ownership renounce: https://etherscan.io/tx/0x9eccc3c1e1bc7aa3596e697943b075308d3249909435a29ec728e73128f5dd12

Auditor Score =85
Audit Passed

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that actagainst the nature of decentralization, such as explicit ownership or specialized access roles incombination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimalEVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on howblock.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functionsbeing invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that mayresult in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to makethe codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code,such as a constructor assignment imposing different require statements on the input variables than a setterfunction.

Coding Best Practices

ERC 20 Conding Standards are a set of rules that each developer should follow to ensure the code meet a set of creterias and is readable by all the developers.

Disclaimer

CFGNINJA has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocation for the Project, and users relying on this report should not consider this as having any merit for financial advice in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided 'as is, and CFGNINJA is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will CFGNINJA or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment services provided by CFGNINJA are subject to dependencies and are under continuing development. You agree that your access or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.