

Smart Contract

Security Assessment

**For The Krypto
Miners Club
20 Feb 2023**



Table of Contents

3	Disclaimer
4	Executive Summary
5	Overview
6	Findings Summary & Legend
8	Manual Review <ul style="list-style-type: none">• Issue Checking Status• Audit Findings• Functional Test Status• Omitted Results
20	Automated Review <ul style="list-style-type: none">• Unified Model Language
22	Conclusion

DISCLAIMER

This independent audit has been conducted to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocacy for the Project that was audited, and users relying on this audit 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 may be pursued by the Project in question, and that 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 completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by the auditor.

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 in this report. Information is provided 'as is', and the auditor is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will the auditor 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 and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Auditor retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Auditor is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. The auditor may, at its discretion, claim bug bounties from third-parties while doing

so.

Executive Summary

Severity	Found
● High	0
● Medium	0
● Low	12
● Informational	18
Total	30

We performed an independent technical audit to identify Smart Contracts uncertainties. This shall protect the code from illegitimate authorization attempts or external & internal threats of any type. This also ensures end-to-end proofing of the contract from frauds. The audit was performed semi-manually. We analyzed the Smart Contracts code line-by-line and used an automation tool to report any suspicious code.

The following tools were used:

- Truffle
- Remix IDE
- Slither

Overview

This report has been prepared for The Krypto Miners Club. This audit provides a user-centered examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

Summary





Project Name	The Krypto Miners Club
Platform	Unpublished
Language	Solidity

Contracts Assessed





Name	Location
TKMC.sol	Not Published
Strings.sol	In TKMC contract
Math.sol	In TKMC contract
Ownable.sol	In TKMC contract
IERC20.sol	In TKMC contract
ReentrancyGuard.sol	In TKMC contract

Name	Location
Address.sol	In TKMC contract
Context.sol	In TKMC contract
IERC721Receiver.sol	In TKMC contract
IERC721Metadata.sol	In TKMC contract
ERC721.sol	In TKMC contract
IERC165.sol	In TKMC contract
ERC165.sol	In TKMC contract

Findings Summary

Severity	Found
 High	0
 Medium	0
 Low	12
 Informational	18
Total	30

Classification of Issues

 High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
 Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
 Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
 Informational	Consistency, syntax or style best practices, Generally pose a negligible level of risk, if any.

Manual Review

A large, solid black diagonal shape that starts from the bottom left and extends towards the top right, covering the lower right portion of the page.

Issues Checking Status

Issue Description	Checking Status
Compiler errors	PASS
Race conditions and Reentrancy. Cross-function race conditions.	PASS
Possible delays in data delivery.	PASS
Oracle calls.	PASS
Front running.	PASS
Timestamp dependence.	PASS
Integer Overflow and Underflow.	PASS
DoS with Revert.	PASS
DoS with block gas limit.	PASS
Methods execution permissions.	PASS
Economy model of the contract.	PASS
The impact of the exchange rate on the logic.	PASS
Private user data leaks.	PASS
Malicious Event log.	PASS
Scoping and Declarations.	PASS
Uninitialized storage pointers.	PASS

Arithmetic accuracy.	PASS
Design Logic.	PASS
Cross-function race conditions.	PASS
Safe Open Zeppelin contracts implementation and usage.	PASS
Fallback function security.	PASS

Audit Findings

Severity	Lowx3
Contract	TKMC.sol
Description	Return value ignored
Code Snippet	<pre>function preSaleMint(uint8 _amount) external callerIsUser { ... USDTR.transferFrom(msg.sender, address(this), preSaleCost * _amount); function mint(uint8 _amount) external callerIsUser { ... USDTR.transferFrom(msg.sender, address(this), publicCost * _amount) function withdraw() external onlyOwner nonReentrant { ... USDTR.transfer(msg.sender, balance)</pre>
Recommendation	The return value of the external/transferFrom call is not checked. If the function does not revert in case of failure, an attacker can call it for free. Make sure the value returns true on success.
Status	

Audit Findings

Severity	Informational
Contract	TKMC.sol
Description	Costly operations inside loop
Code Snippet	<pre>function setAllowlistAddresses(address[] calldata _users) external onlyOwner { for (uint256 i = 0; i < _users.length; i++) { _allowList[_users[i]] = true; } }</pre>
Recommendation	Storing addresses in the contract is extremely gas-costly. If the whitelist is long, it may be worth it to consider implementing a merkle tree.
Status	

Audit Findings

Severity	Informational
Contract	TKMC.sol
Description	Hardcoded value
Code Snippet	IERC20 public USDTR = IERC20(...)
Recommendation	Hardcoding the token address for the pay token without including a set function will make it impossible to change the pay token (if ever necessary) without redeploying. It is advisable to include a set function that will allow the owner to change this address to allow for greater flexibility.
Status	

Functional Test Status

Function Name	Type/Return Type	Score
Math		
average	internal	PASS
ceilDiv	internal	PASS
log10	internal	PASS
log2	internal	PASS
log256	internal	PASS
max	internal	PASS
min	internal	PASS
mulDiv	internal	PASS
sqrt	internal	PASS
IERC20		
allowance	read/external	PASS
approve	write/external	PASS
balanceOf	read/external	PASS
totalSupply	read/external	PASS
transfer	write/external	PASS
transferFrom	write/external	PASS
Context		

Function Name	Type/Return Type	Score
_msgData	internal	PASS
_msgSender	internal	PASS
Ownable		
_checkOwner	internal	PASS
transferOwnership	write/public	PASS
owner	read/public	PASS
renounceOwnership	write/public	PASS
Address		
functionCall	internal	PASS
functionCallWithValue	internal	PASS
functionDelegateCall	internal	PASS
functionStaticCall	internal	PASS
isContract	internal	PASS
sendValue	internal	PASS
verifyCallResult	internal	PASS
IERC721Metadata		
name	read/external	PASS
symbol	read/external	PASS
tokenURI	read/external	PASS

IERC721Receiver		
onERC721Received	external	PASS
IERC721		
approve	write/external	PASS
balanceOf	read/external	PASS
getApproved	read/external	PASS
isApprovedForAll	read/external	PASS
ownerOf	read/external	PASS
safeTransferFrom	write/external	PASS
setApprovalForAll	write/external	PASS
transferFrom	write/external	PASS
ERC721		
_afterTokenTransfer	internal	PASS
_approve	internal	PASS
_baseURI	internal	PASS
_beforeTokenTransfer	internal	PASS
_burn	internal	PASS
_checkOnERC721Received	private	PASS
_exists	internal	PASS
_isApprovedOrOwner	internal	PASS
_mint	internal	PASS
_ownerOf	internal	PASS
_requireMinted	internal	PASS

_safeMint	internal	PASS
_safeTransfer	internal	PASS
_setApprovalForAll	internal	PASS
_transfer	internal	PASS
approve	public	PASS
balanceOf	public	PASS
constructor	public	PASS
getApproved	public	PASS
isApprovedForAll	public	PASS
name	public	PASS
ownerOf	public	PASS
safeTransferFrom	public	PASS
setApprovalForAll	public	PASS
supportsInterface	public	PASS
symbol	public	PASS
tokenURI	public	PASS
transferFrom	public	PASS
TKMC		
_baseURI	internal	PASS
airdrop	write/external	PASS
constructor	public	PASS
isAllowlisted	read/external	PASS

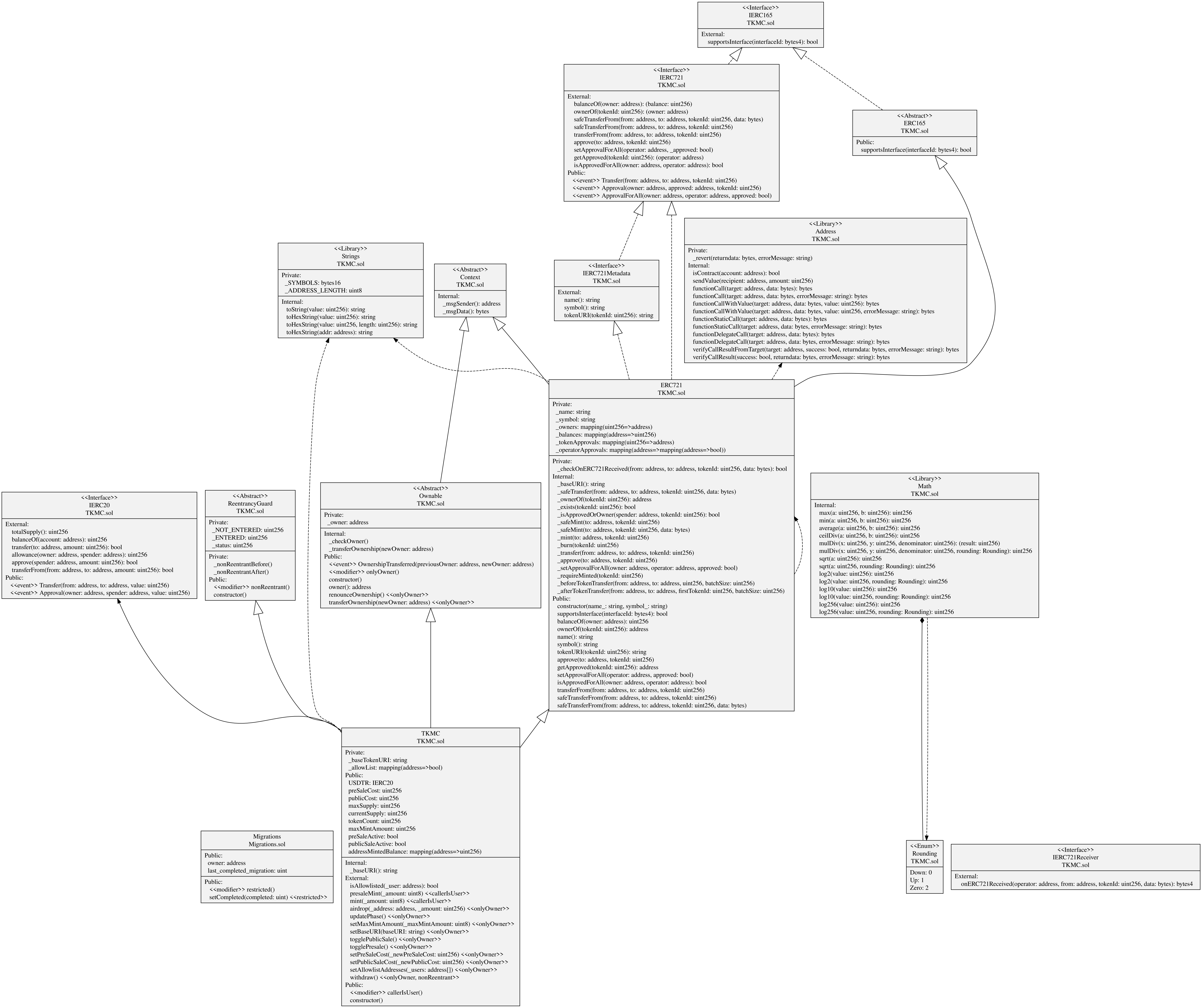
mint	write/external	PASS
presaleMint	write/external	PASS
setAllowlistAddresses	write/external	PASS
setBaseURI	write/external	PASS
setMaxMintAmount	write/external	PASS
setPresaleCost	write/external	PASS
setPublicSaleCost	write/external	PASS
togglePresale	write/external	PASS
togglePublicSale	write/external	PASS
updatePhase	write/external	PASS
withdraw	write/external	PASS
ReentrancyGuard		
_nonReentrantAfter	private	PASS
_nonReentrantBefore	private	PASS

Omitted Results

Note: Any issues that have been omitted from this report have been deemed by the reviewing team as irrelevant, inapplicable, and/or negligible to the proper functioning of this contract. Thus, any omitted issues can be safely ignored.

Automated Review





Conclusion

The smart contracts reviewed in this audit contain no critical severity issues and all Medium to Low issues have either been corrected or acknowledged.

Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.

