



**DESSERT**  
FINANCE

**Kill Kount (KKT)**

**BEP-20 Audit**

Performed at block **SAMPLE**

PERFORMED BY DESSERT FINANCE

FOR CONTRACT ADDRESS: 0x80976310B15043e746ef3e0Bfb47584bcdCd35f0  
FOR CONTRACT ADDRESS: 0x2A4E5876ccD58a05cab3048EdF9C38bAa4745Fb9  
FOR CONTRACT ADDRESS: 0xA03fb4e55C78aE9b772E01A4a9873Db670862757

## INITIAL DISCLAIMER

Dessert Finance provides due-diligence project audits for various projects. Dessert Finance in no way guarantees that a project will not remove liquidity, sell off team supply, or otherwise exit scam.

Dessert Finance does the legwork and provides public information about the project in an easy-to-understand format for the common person.

Agreeing to an audit in no way guarantees that a team will not remove *all* liquidity (“Rug Pull”), remove liquidity slowly, sell off tokens, quit the project, or completely exit scam. There is also no way to prevent private sale holders from selling off their tokens. It is ultimately your responsibility to read through all documentation, social media posts, and contract code of each individual project to draw your own conclusions and set your own risk tolerance.

Dessert Finance in no way takes responsibility for any losses, nor does Dessert Finance encourage any speculative investments. The information provided in this audit is for information purposes only and should not be considered investment advice. Dessert Finance does not endorse, recommend, support, or suggest any projects that have been audited. An audit is an informational report based on our findings, We recommend you do your own research, we will never endorse any project to invest in.

# Table of Contents



1. Contract Code Audit – Token Overview
2. BEP-20 Contract Code Audit – Overview
3. BEP-20 Contract Code Audit – Vulnerabilities Checked
4. Contract Code Audit – Contract Ownership
5. Contract Code Audit – Owner Accessible Functions
6. Liquidity Ownership – Locked / Unlocked
7. Contract Code Audit – Mint Functions
8. Contract Transaction Fees
9. Website Overview
10. Social Media
11. Top Token Holders/Wallets
12. Location Audit
13. Review of Team
14. Roadmap
15. Disclaimers

# Contract Code Audit – Token Overview





# BEP-20 Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on Kill Kount (KKT)

```
pragma solidity ^0.8.6;
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/token/ERC20/Address.sol";
import "@openzeppelin/contracts/token/ERC20/Token.sol";
import "@openzeppelin/contracts/token/ERC20/TokenHolder.sol";
import "@openzeppelin/contracts/token/ERC20/TokenHolder.sol";
import "@openzeppelin/contracts/token/ERC20/TokenHolder.sol";
import "@openzeppelin/contracts/token/ERC20/TokenHolder.sol";
import "@openzeppelin/contracts/token/ERC20/TokenHolder.sol";

contract KillKount is ERC20, Address, TokenHolder, TokenHolder {
    using SafeMath for uint256;
    using Address for address;
    using Address for address payable;

    struct Fees {
        uint256 liquidity;
        uint256 marketing;
        uint256 tokenReflection;
        uint256 buyback;
        uint256 divisor;
    }

    struct TokenTracker {
        uint256 liquidity;
        uint256 marketingTokens;
        uint256 reward;
        uint256 buyback;
    }

    mapping (address => uint256) private balances;
    mapping (address => mapping (address => uint256)) private allowances;

    mapping (address => bool) public _isExcludedFromFee;
    mapping (address => bool) public _isWhitelist;
    mapping (address => bool) private _automatedMarketMakerPairs;

    string private constant _name = "Kill Kount";
    string private constant _symbol = "KKT";
    uint256 private constant _decimals = 18;
    uint256 private constant _totalSupply = 11 * 10 ** 6 * 10 ** _decimals;

    // Trackers for various pending token swaps and fees
    Fees public buyFees;
    Fees public sellFees;
    Fees public transferFees;
    TokenTracker public tokenTracker;

    uint256 public gasForProcessing = 200000;

    address public marketingMallot;
    address public buybackVault;
    IDynamicReflector public reflectorContract;

    bool private _initComplete;

    constructor (address routerAddress, address tokenOwner, address _marketingMallot) IPushSupport (token) payable {
        super(routerAddress, tokenOwner, _marketingMallot);
        balances[tokenOwner] = _totalSupply;

        marketingMallot = _marketingMallot;

        _totalSupply = _totalSupply * 10 ** 1;

        buyFees = Fees({
            liquidity: 0
        });
    }
}
```

## Contract Address

0x80976310B15043e746ef3e0Bfb47584bcdCd35f0

## TokenTracker

Kill Kount (KKT)

## Contract Creator

0xb38e4a0569e89e4ce90995f1a49d211f99875ed9

## Source Code

Contract Source Code Verified

## Contract Name

KillKount

## Other Settings

default evmVersion

## Compiler Version

v0.8.6+commit.11564f7e

## Optimization Enabled

Yes with 100 runs

Code is truncated to fit the constraints of this document.

[The code in its entirety can be viewed here.](#)

# BEP-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Compiler Errors	Complete	Complete	✓ Low / No Risk
Outdated Compiler Version	Complete	Complete	✓ Low / No Risk
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

The contract code is verified on BSCScan.

The vulnerabilities listed above were not found in the token's Smart Contract.

# Contract Code Audit – Contract Ownership

Contract Ownership has not been renounced at the time of Audit



The contract ownership is not currently renounced.

We have placed the contract owner address below for your viewing:

[0xff6d35088a8748475e6fc508547ec9279f3e513b](https://www.etherscan.io/address/0xff6d35088a8748475e6fc508547ec9279f3e513b)

The address above has authority over the ownable functions within the contract.

This allows the owner to call certain functions within the contract. Any compromise to the owner wallet may allow these privileges to be exploited.

We recommend:

- Establishing a Time-Lock with reasonable latency
- Assignment of privileged roles to multi-signature wallets

# Contract Code Audit – Owner Accessible Functions

Function Name	Parameters	Visibility	Audit Notes
setMarketingWallet	address _marketingWallet	public	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
setVaultAddress	address _buybackVault	public	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
excludeFromFee	address account, bool shouldExclude	public	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
updateBuyFees	uint256 reflectionFee, uint256 liquidityFee, uint256 marketingFee, uint256 newFeeDivisor, uint256 buybackFee	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
updateSellFees	uint256 reflectionFee, uint256 liquidityFee, uint256 marketingFee, uint256 newFeeDivisor, uint256 buybackFee	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
updateTransferFees	uint256 reflectionFee, uint256 liquidityFee, uint256 marketingFee, uint256 newFeeDivisor, uint256 buybackFee	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
authorizeCaller	address authAddress, bool shouldAuthorize	external override	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
updateLPPair	address newAddress	public override	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
updateReflectionContract	address newReflectorAddress	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
excludeFromRewards	address userAddress, bool shouldExclude	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
updateWhitelist	address wallet, bool shouldWhitelist	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
batchUpdateWhitelist	address[] memory wallets, bool shouldWhitelist	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.

The functions listed above can be called by the contract owner.

If contract ownership has been renounced there is no way for the above listed functions to be called.



# Liquidity Ownership – Locked / Unlocked

No locked liquidity information has been found.



This page will contain links to locked liquidity for the project if we are able to locate that information. Locked liquidity information was not found on the project's website.

# Contract Code Audit – Mint Functions

This Contract Cannot Mint New KKT Tokens.



We do understand that sometimes mint functions are essential to the functionality of the project.

**A mint function was not found in the contract code.**

# Contract Transaction Fees

At the time of Audit the transaction fees ("tax") listed below are the fees associated with trading. These fees are taken from every buy and sell transaction unless otherwise stated.

Kill Kount Token (KKT) is a Binance Smart Chain token that distributes passive earning of MTX to holders and serves as the in-game currency for the Kill Kount series of games. KKT will be offering a 0% buy tax and 15% sell tax. KKT has designed an innovative mechanism - "Killshot". The Killshot function will accumulate in BNB from every sell transaction. All buy back tokens will be redistributed through the ecosystem. The buyback will happen at various times but will be announced in advance to the KKT community.

## Marketing

4% of every sell will be accumulated in a separate marketing wallet to assist with on going marketing, gaming and partnership expenses.

**4%**

## Liquidity

6% of every sell will be converted to BNB and sent directly to the liquidity pool to promote price and chart stability.

**6%**

## MTX Rewards

Automatic MTX reflections will be received by holding KKT. 2% of every sell will be given to holders automatically.

**2%**

## Buyback

There will be scheduled "Killshot" buyback events. 3% of every sell will be accumulated in BNB and will be used as a scheduled buy back.

**3%**



# BEP-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Compiler Errors	Complete	Complete	✓ Low / No Risk
Outdated Compiler Version	Complete	Complete	✓ Low / No Risk
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

The contract code is verified on BSCScan.

The vulnerabilities listed above were not found in the token's Smart Contract.



# Contract Code Audit – Owner Accessible Functions

Function Name	Parameters	Visibility	Audit Notes
updateShare	address shareholder, uint256 amount, bool shouldProcess	external override	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
setShares	address sendingShareholder, uint256 senderBalance, address receivingShareholder, uint256 receiverBalance	public override	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
deposit		external payable override	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
renounceOwnership		public virtual	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
transferOwnership	address newOwner	public virtual	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
lock	uint256 time	public virtual	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.

The functions listed above can be called by the contract owner.

If contract ownership has been renounced there is no way for the above listed functions to be called.

# Contract Code Audit – Authorized Accessible Functions

Function Name	Parameters	Audit Notes
excludeFromReward	address shareholder, bool shouldExclude	Authorized user can call this function.
setDistributionCriteria	uint256 _minPeriod, uint256 _minDistribution	Authorized user can call this function.
setRewardToCurrency	bool andSwap	Authorized user can call this function.
setRewardToToken	address _tokenAddress, bool andSwap	Authorized user can call this function.
claimDividendFor	address shareholder	Authorized user can call this function.
authorizeByAuthorized	address authAddress	Authorized user can call this function.
renounceAuthorization		Authorized user can call this function.

The functions listed above can be called by authorized users.

If contract ownership has been renounced there is no way for the above listed functions to be called.

# BEP-20 Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on KillKountBuybackVault

```
pragma solidity ^4.8;
import './utils/IFlowSupport.sol';
import './interfaces/ISupportTokenFunction.sol';

contract KillKountBuybackVault is IFlowSupport, ISupportTokenFunction {
    using SafeMath for uint256;

    struct TokenInfo {
        string name;
        uint256 decimals;
    }

    address public killKountAddress;
    uint256 private baseDecimals = 10 ** 18;
    TokenInfo private kkt;

    constructor(address _routerAddress, address contractOwner, address tokenAddress) IFlowSupport {
        super(_routerAddress, contractOwner);
        killKountAddress = tokenAddress;

        IERC20 _token = IERC20(tokenAddress);
        kkt.name = _token.name();
        kkt.decimals = 10 ** uint256(_token.decimals());

        _owner = contractOwner;
        authorizeCaller[contractOwner] = true;
    }

    receive() external payable {}
    fallback() external payable {}

    function killKountInfo() public view returns (string memory name, uint256 wholeKKBalance, uint256 KKBalance) {
        name = kkt.name;
        KKBalance = IERC20(killKountAddress).balanceOf(address(this));
        wholeKKBalance = KKBalance * kkt.decimals / baseDecimals;
    }

    function buyBackToken(address tokenAddress) external onlyOwner {
        require(tokenAddress != killKountAddress, "Cannot buy back with own KKT");

        IERC20 _token = IERC20(tokenAddress);
        uint256 balance = _token.balanceOf(address(this));
        _token.transfer(msg.sender(), balance);
    }

    function buyBack() public authorized {
        buyBackToken(address(this), balance);
    }

    function buyBack(uint256 buyBackAmountInWholeKKB) public authorized {
        buyBackToken(buyBackAmountInWholeKKB * baseDecimals);
    }

    function buyBack(uint256 buyBackAmount) public authorized {
        buyBack(buyBackAmount);
    }

    function buyBackToken(uint256 tokensForLiquidityInWholeTokens, uint256 tokensForMarketingInWholeTokens, uint256 tokensForReflection
        uint256 decimals = kkt.decimals;
        uint256 tokensForLiquidity = tokensForLiquidityInWholeTokens * decimals;
        uint256 tokensForMarketing = tokensForMarketingInWholeTokens * decimals;
        uint256 tokensForReflection = tokensForReflectionInWholeTokens * decimals;
        uint256 tokensToBuyBack = tokensForLiquidity + tokensForMarketing + tokensForReflection;
    }
}
```

## Contract Address

0xA03fb4e55C78aE9b772E01A4a9873Db670862757

## Contract Creator

0xb38e4a0569e89e4ce90995f1a49d211f99875ed9

## Source Code

Contract Source Code Verified

## Contract Name

KillKountBuybackVault

## Compiler Version

v0.8.6+commit.11564f7e

## Optimization Enabled

Yes with 100 runs

Code is truncated to fit the constraints of this document.

[The code in its entirety can be viewed here.](#)

# BEP-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Compiler Errors	Complete	Complete	✓ Low / No Risk
Outdated Compiler Version	Complete	Complete	✓ Low / No Risk
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

The contract code is verified on BSCScan.

The vulnerabilities listed above were not found in the token's Smart Contract.

# Contract Code Audit – Owner Accessible Functions

Function Name	Parameters	Visibility	Audit Notes
clearStuckTokens	address tokenAddress	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
renounceOwnership		public virtual	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
transferOwnership	address newOwner	public virtual	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
lock	uint256 time	public virtual	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.

The functions listed above can be called by the contract owner.

If contract ownership has been renounced there is no way for the above listed functions to be called.



# Contract Code Audit – Authorized Accessible Functions

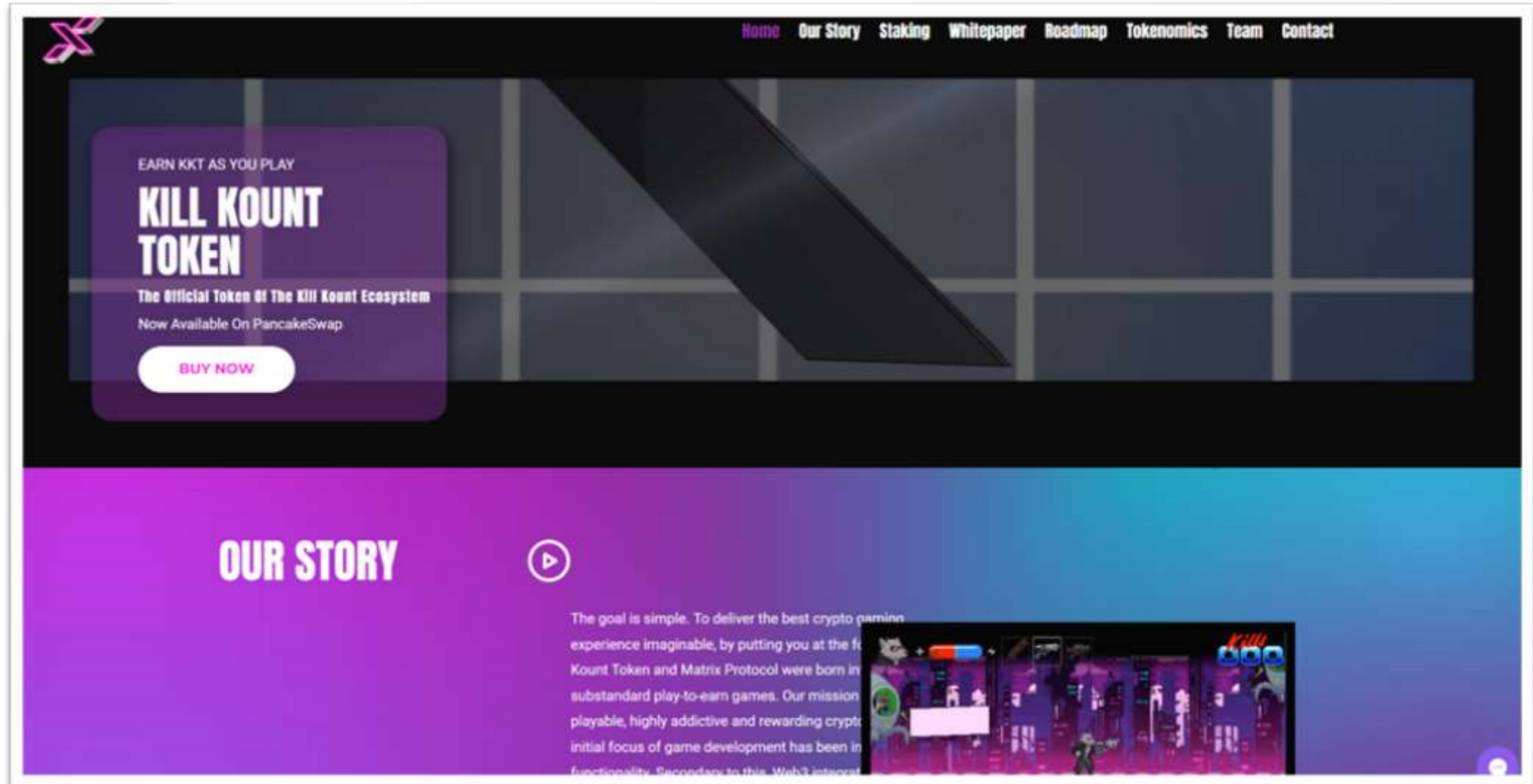
Function Name	Parameters	Audit Notes
buybackMax		Authorized user can call this function.
buyback	uint256 buybackAmountInWholeBNB	Authorized user can call this function.
buybackAdvanced	uint256 buybackAmount	Authorized user can call this function.
injectTokens	uint256 tokensForLiquidityInWholeTokens, uint256 tokensForMarketingInWholeTokens, uint256 tokensForReflectionInWholeTokens, uint256 tokensForBuybackInWholeTokens	Authorized user can call this function.
injectTokensAdvanced	uint256 tokensForLiquidity, uint256 tokensForMarketing, uint256 tokensForReflection, uint256 tokensForBuyback	Authorized user can call this function.
authorizeByAuthorized	address authAddress	Authorized user can call this function.
renounceAuthorization		Authorized user can call this function.

The functions listed above can be called by authorized users.

If contract ownership has been renounced there is no way for the above listed functions to be called.

# Website Part 1 – Overview

## [www.killkounttoken.io](http://www.killkounttoken.io)



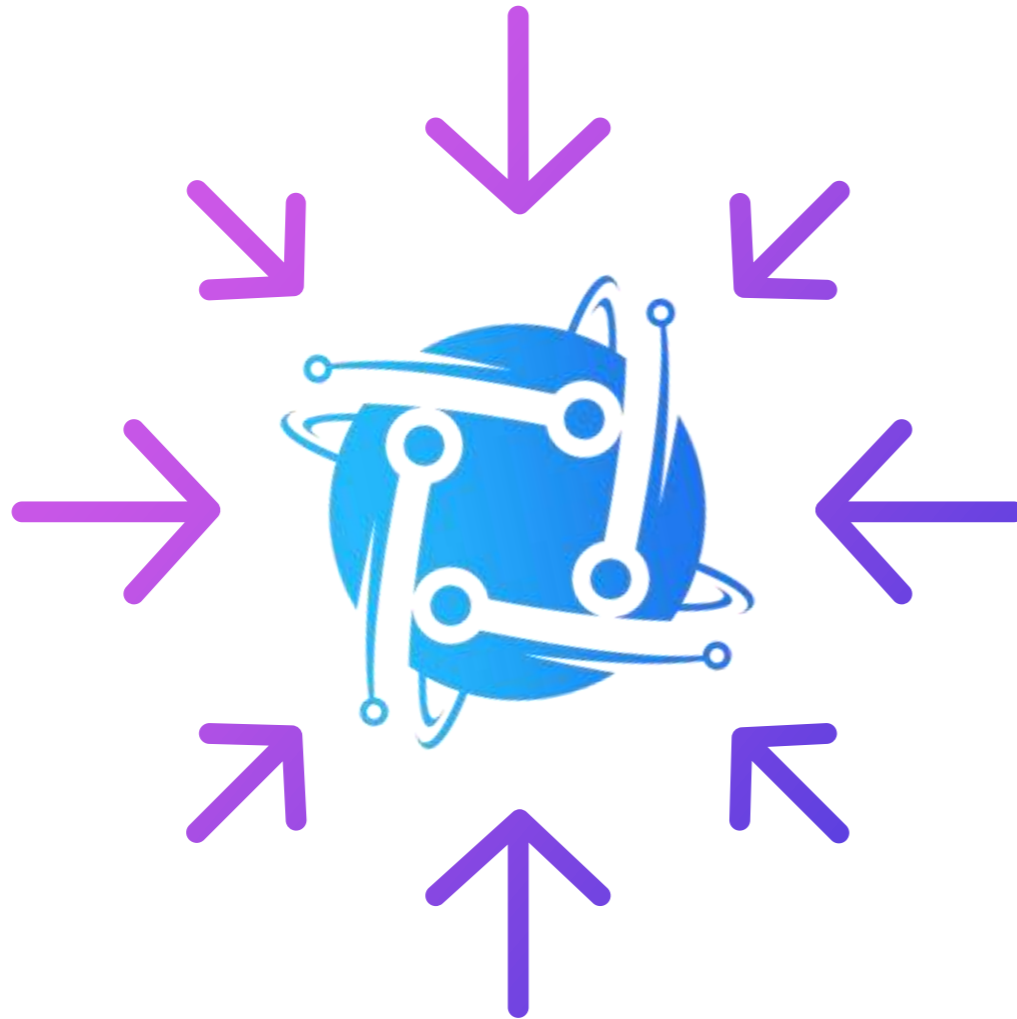
Above images are actual snapshots of the current live website of the project.

Website was registered on 04/03/2022, registration expires 04/03/2025.

✓ This meets the 3 year minimum we like to see on new projects.



## Website Part 2 – Checklist



- ✓ Mobile Friendly
- ✓ No JavaScript Errors
- ✓ Spell Check
- ✓ SSL Certificate

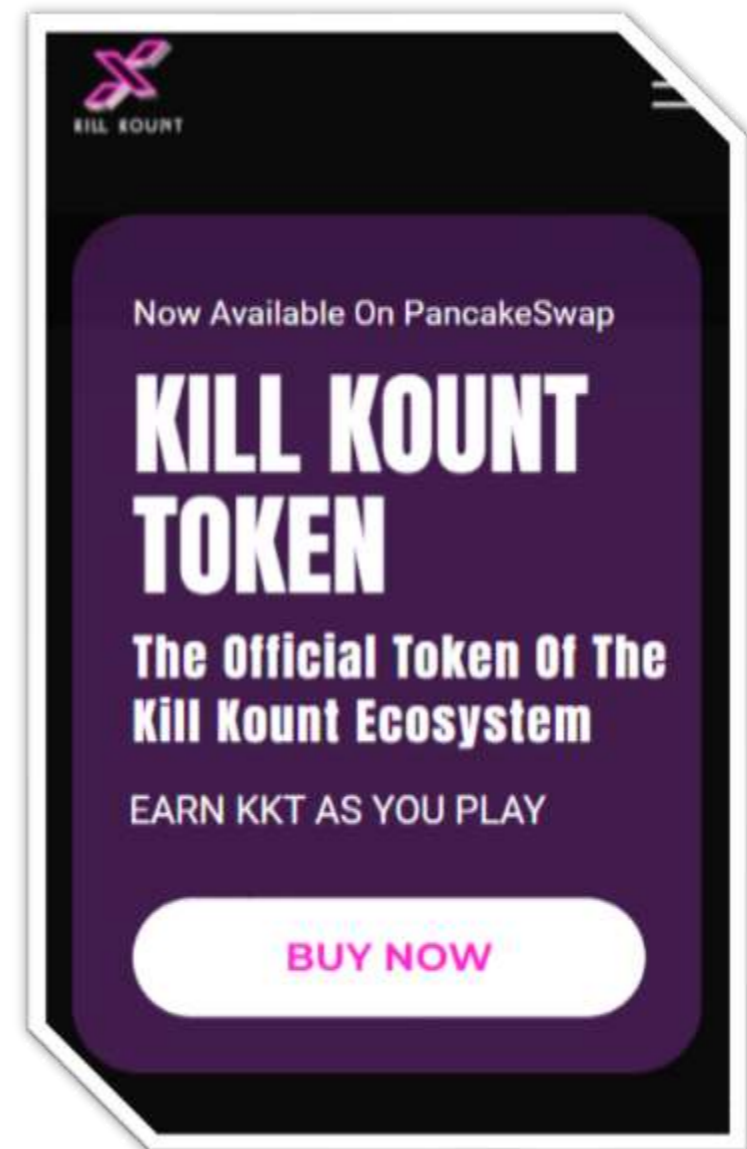
The website contained no JavaScript errors. No typos, or grammatical errors were present, and we found a valid SSL certificate allowing for access via https.

No additional issues were found on the website.

# Website Part 3 – Responsive HTML5 & CSS3

No issues were found on the Mobile Friendly check for the website. All elements loaded properly and browser resize was not an issue. The team has put a considerable amount of thought and effort into making sure their website looks great on all screens.

No severe JavaScript errors were found. No issues with loading elements, code, or stylesheets.



# Website Part 4 (GWS) – General Web Security



## SSL CERTIFICATE

A valid SSL certificate was found. Details are as follows:

Offered to: killcounttoken.io

Issued by: R3

Valid Until: 07/20/2022



## CONTACT EMAIL

A valid contact email was found on the official website. Contact email is listed as shown below:

[Contact](#)

**Live Chat on Website**



## SPAM / MALWARE / POPUPS

No malware found

No injected spam found

No internal server errors

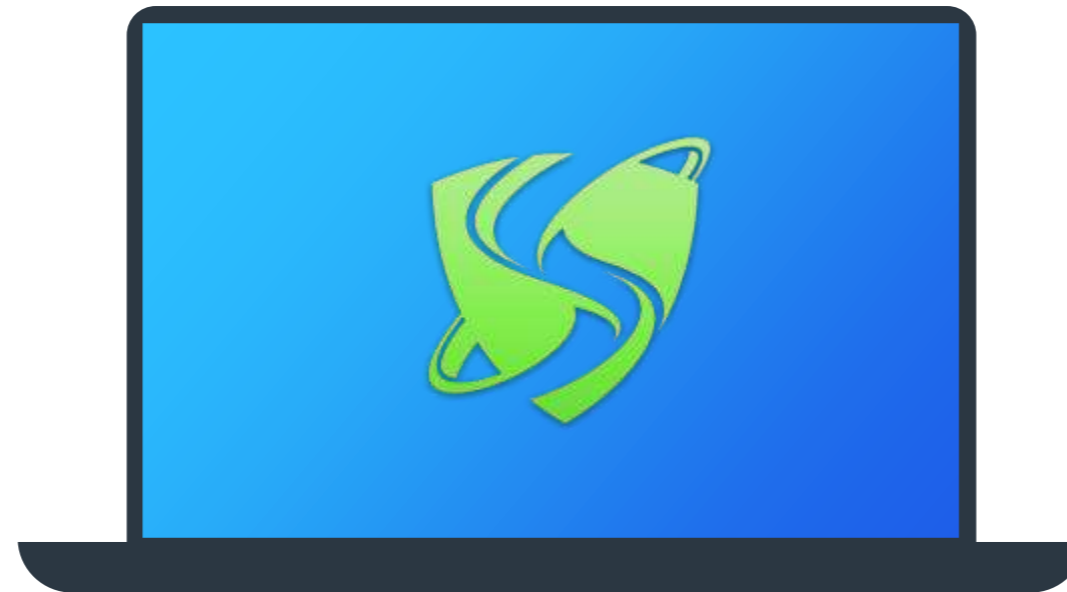
No popups found

Domain is marked clean by Google, McAfee, Sucuri Labs, & ESET





# Social Media



We were able to locate a variety of Social Media networks for the project.

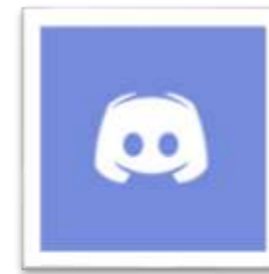
All links have been conveniently placed below.



[Twitter](#)



[Telegram](#)



[Discord](#)

✓ **At least 3 social media networks were found.**

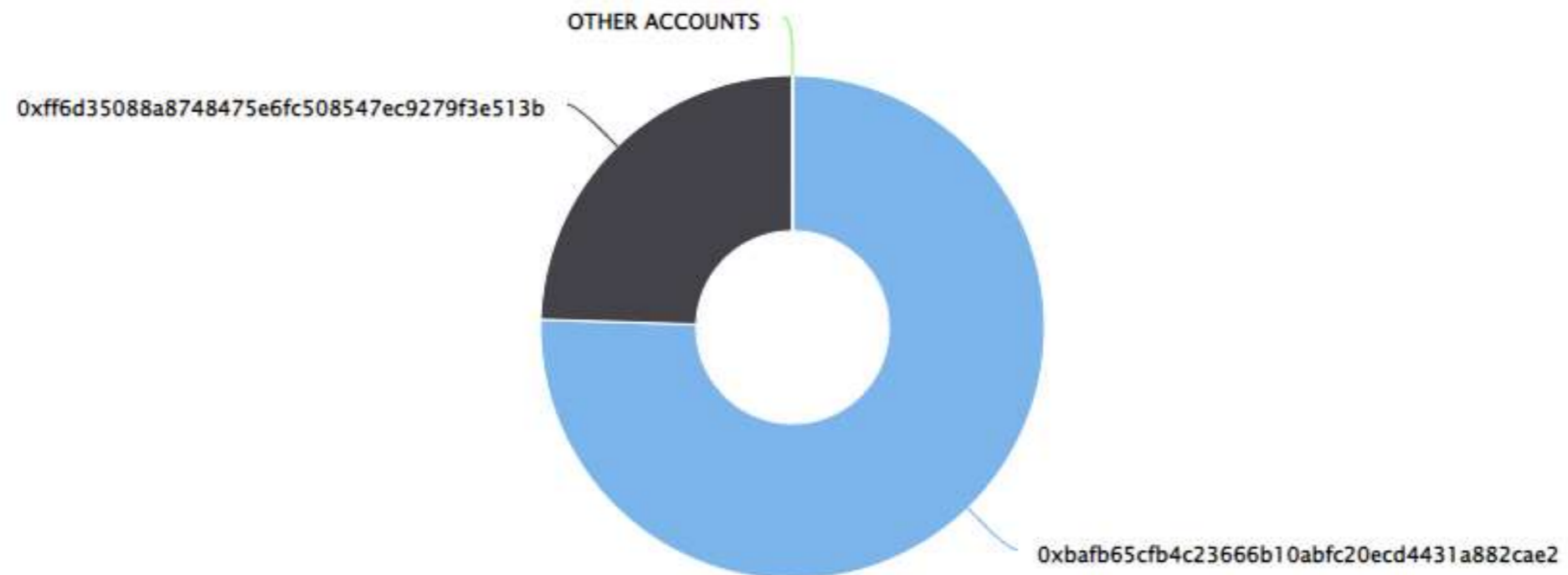
# Top Token Holders

The entire supply was in one wallet at the time of audit. We expect this to change as the project goes through initial distribution phases. Please use the link below to view the most up-to-date holder information.

[Click here to view the most up-to-date list of holders](#)

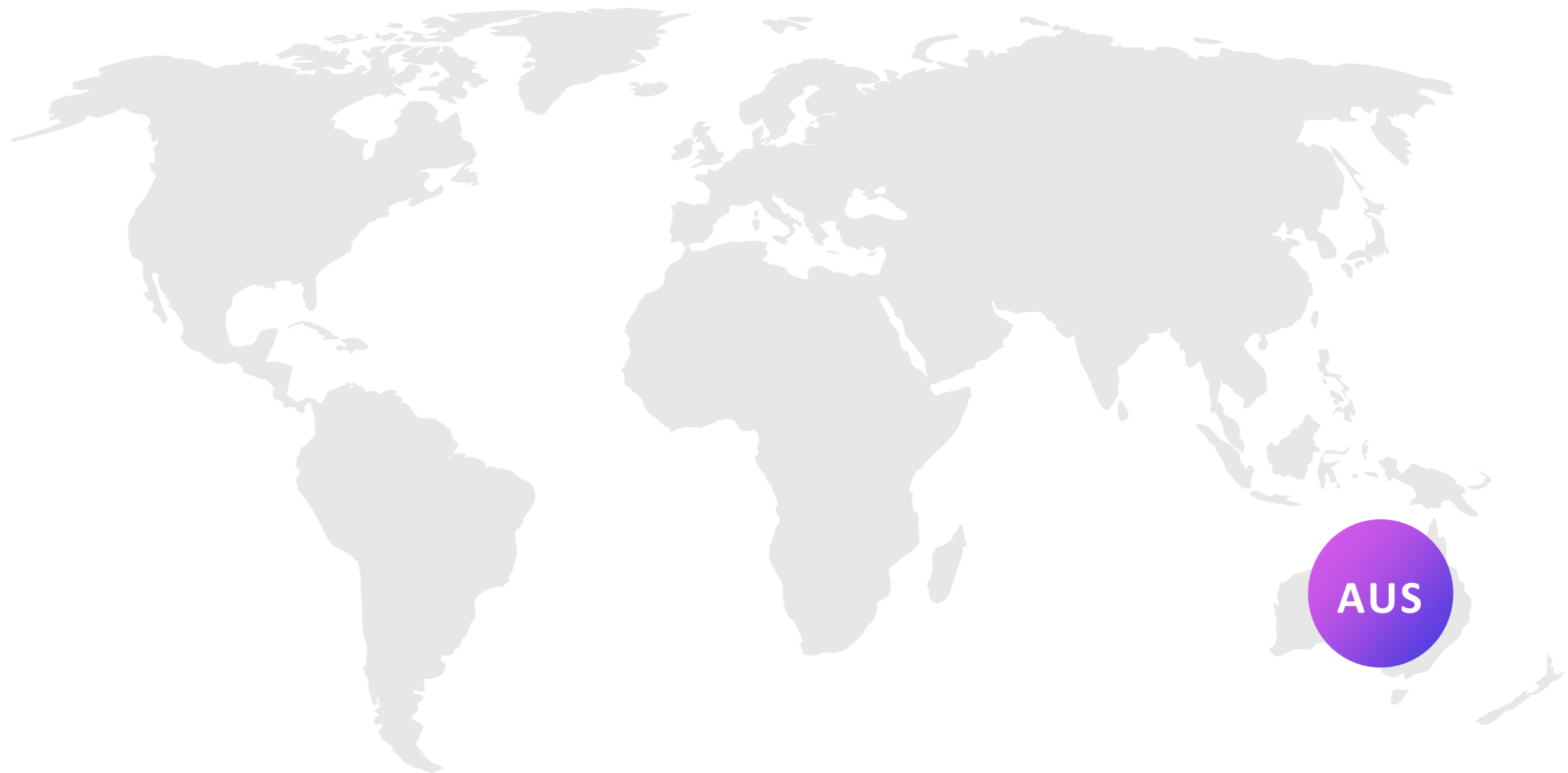
## Kill Kount Top 100 Token Holders

Source: BscScan.com



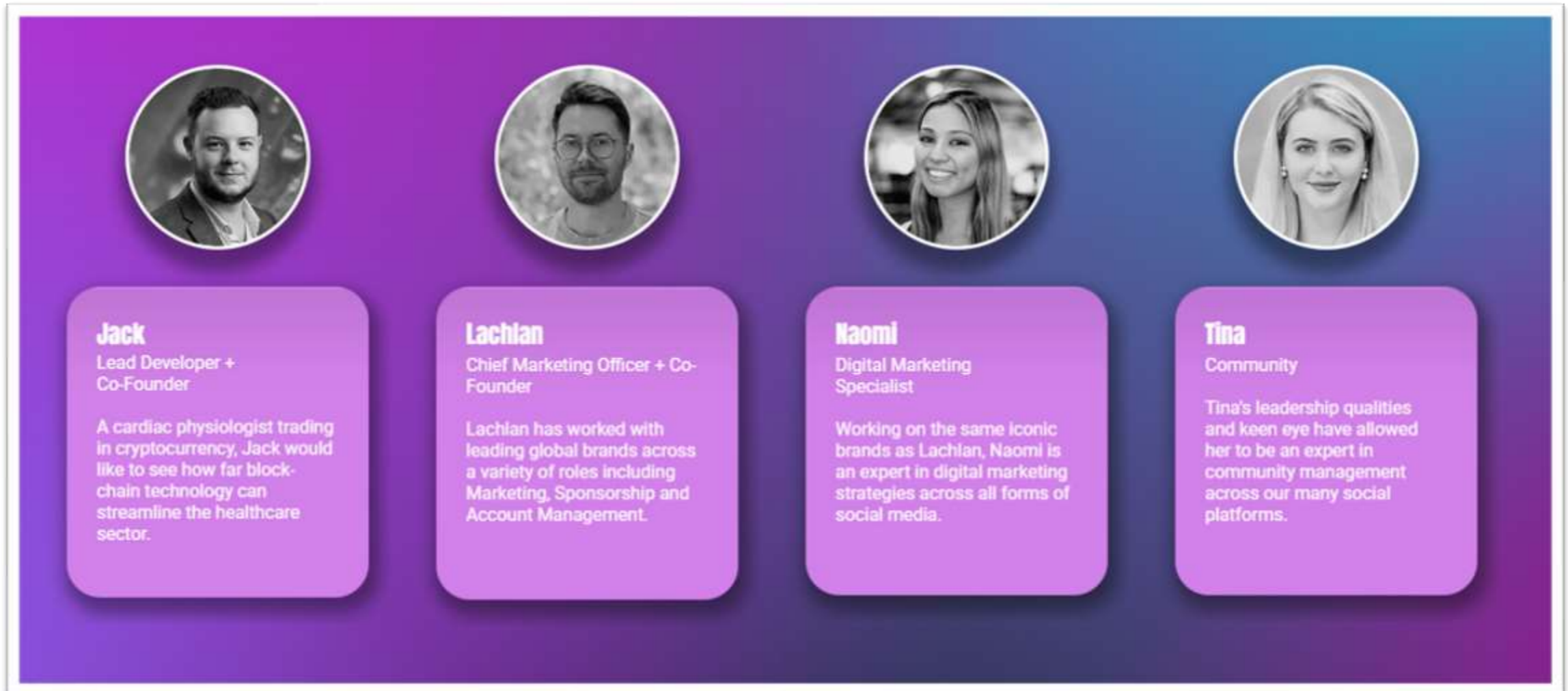
# Location Audit





The primary location of the team is Australia



# Team Overview

The following information about the team has been located on the projects website.



 <p><b>Jack</b> Lead Developer + Co-Founder</p> <p>A cardiac physiologist trading in cryptocurrency, Jack would like to see how far block-chain technology can streamline the healthcare sector.</p>	 <p><b>Lachlan</b> Chief Marketing Officer + Co- Founder</p> <p>Lachlan has worked with leading global brands across a variety of roles including Marketing, Sponsorship and Account Management.</p>	 <p><b>Naomi</b> Digital Marketing Specialist</p> <p>Working on the same iconic brands as Lachlan, Naomi is an expert in digital marketing strategies across all forms of social media.</p>	 <p><b>Tina</b> Community</p> <p>Tina's leadership qualities and keen eye have allowed her to be an expert in community management across our many social platforms.</p>
---	---	--	---

# Roadmap

*A roadmap was found on the official website, we have conveniently placed it on this page for your viewing.*





# Disclaimer



The opinions expressed in this document are for general informational purposes only and are **not intended to provide specific advice or recommendations for any individual or on any specific investment**. It is only intended to provide education and public knowledge regarding projects. This audit is only applied to the type of auditing specified in this report and the scope of given in the results. Other unknown security vulnerabilities are beyond responsibility. Dessert Finance only issues this report based on the attacks or vulnerabilities that already existed or occurred before the issuance of this report. For the emergence of new attacks or vulnerabilities that exist or occur in the future, Dessert Finance lacks the capability to judge its possible impact on the security status of smart contracts, thus taking no responsibility for them. The smart contract analysis and other contents of this report are based solely on the documents and materials that the contract provider has provided to Dessert Finance or was publicly available before the issuance of this report (issuance of report recorded via block number on cover page), if the documents and materials provided by the contract provider are missing, tampered, deleted, concealed or reflected in a situation that is inconsistent with the actual situation, or if the documents and materials provided are changed after the issuance of this report, Dessert Finance assumes no responsibility for the resulting loss or adverse effects. Due to the technical limitations of any organization, this report conducted by Dessert Finance still has the possibility that the entire risk cannot be completely detected. Dessert Finance disclaims any liability for the resulting losses.

Dessert Finance provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Even projects with a low risk score have been known to pull liquidity, sell all team tokens, or exit-scam. Please exercise caution when dealing with any cryptocurrency related platforms.

The final interpretation of this statement belongs to Dessert Finance.

Dessert Finance highly advises against using cryptocurrencies as speculative investments and they should be used solely for the utility they aim to provide.



# Thank You

DESSERT FINANCE PROJECT AUDIT HAS BEEN COMPLETED FOR KILL KOUNT (KKT) AT BLOCK NUMBER: **17524452**

THIS AUDIT IS ONLY VALID IF VIEWED ON [HTTPS://WWW.DSSERTSWAP.FINANCE](https://www.dessertswap.finance)

[www.dessertswap.finance](http://www.dessertswap.finance)  
<https://t.me/dessertswap>