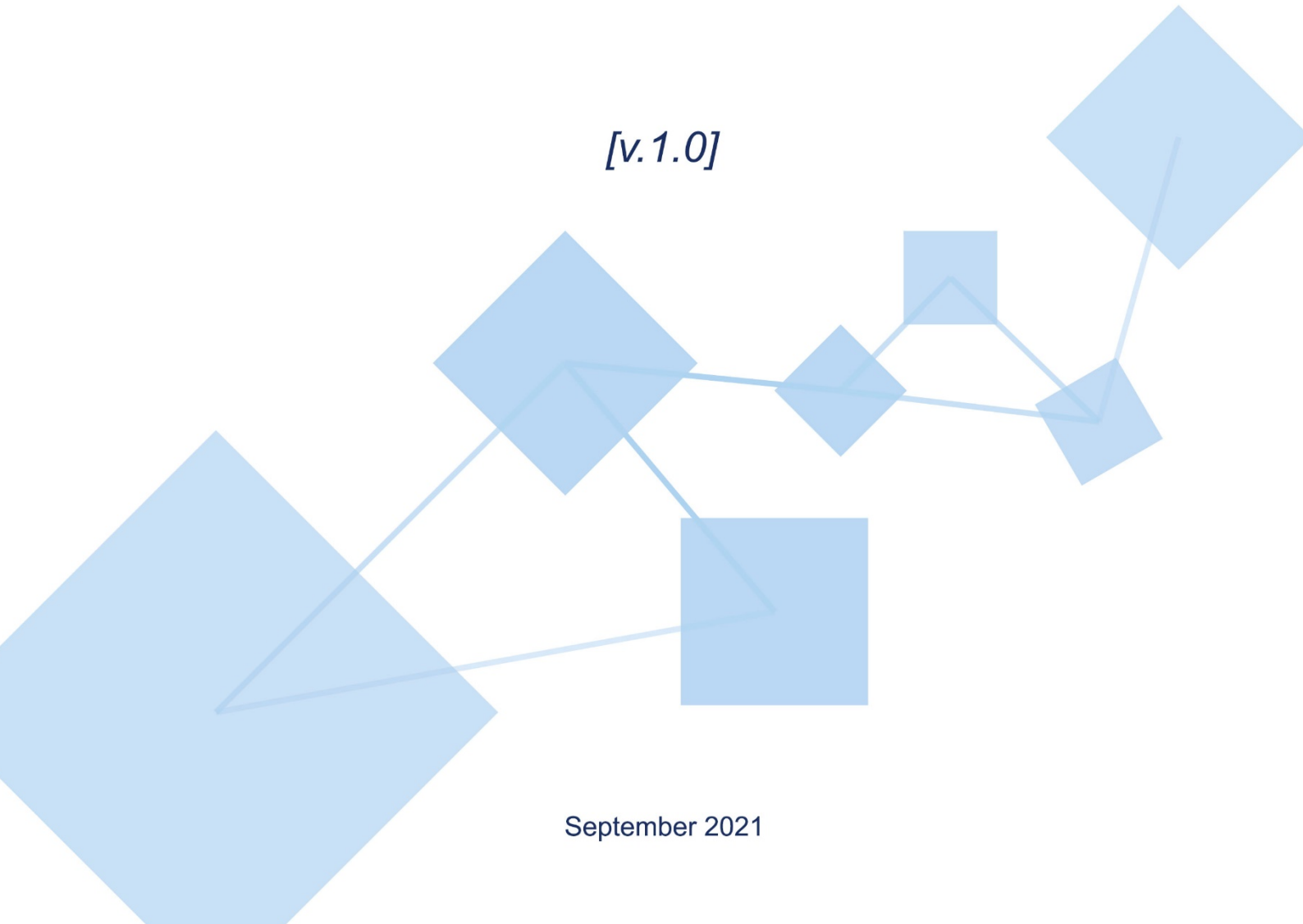
CheckPoint

Token Security Audit Report
Prepared for Wraith

[v.1.0]



September 2021

Document Properties

Client	Wraith
Platform	Binance Smart Chain
Language	Solidity
Codebase	0xE0579e00DcbF6D5a3b9C22DAD4741594acA10c80

Audit Summary

Delivery Date	03.09.2021
Audit Methodology	Static Analysis, Manual Review
Auditor(s)	Erno Patiala
Classification	Publlic
Version	1.0

Contact Information

Company	CheckPoint
Name	Hanna Järvinen
Telegram	t.me/checkpointreport
E-mail	contact@checkpoint.report

Remark: For more information about this document and its contents, please contact CheckPoint team

Table Of Contents

1 Executive Summary

On 03/09/2021, CheckPoint conducted a full audit for the Wraith to verify the overall security posture including a smart contract review to discover issues and vulnerabilities in the source code. Static Code Analysis, Dynamic Analysis, and Manual Review werdone in conjunction to identify smart contract vulnerabilities together with technical & business logic flaws that may be exposed to the potential risk of the platform and the ecosystem.

After further analysis and internal discussion, we determined a few issues of varying severities that need to be brought up and paid more attention to. More information can be found in **Section 5 'Audit Result'**. Practical recommendations are provided according to each vulnerability found and should be followed to remediate the issue.

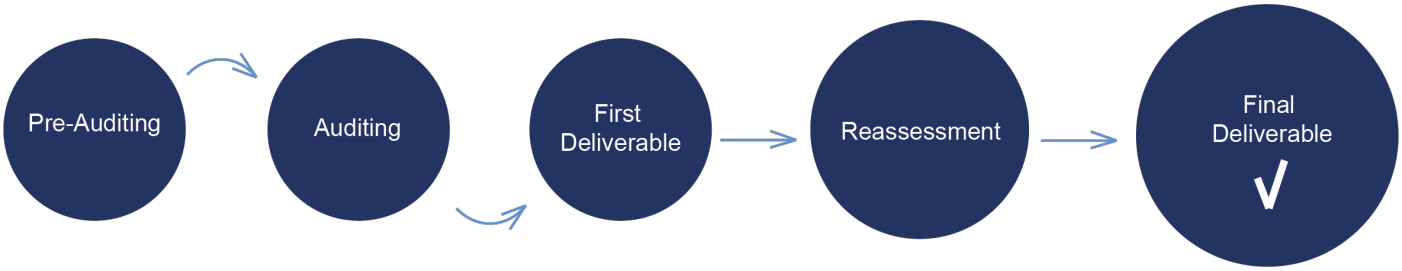
Wraith

Medium Risk Level

Communication Channels	Website Content Analysis, Social Media Listening
Smart Contract Code	Smart Contract Details, Contract Function Details, Issues Checking Status, Detailed Findings Information

THIS TOKEN PASSES CHECKPOINT'S SECURITY VERIFICATION STANDART

2 Audit Methodology



CheckPoint conducts the following procedure to enhance the security level of our clients' tokens:

- **Pre-Auditing**

  Planning a comprehensive survey of the token, its ecosystem, possible risks & prospects, getting to understand the overall operations of the related smart contracts, checking for readiness, and preparing for the auditing.

- **Auditing**

  Study of all available information about the token on the Web, inspecting the smart contracts using automated analysis tools and manual analysis by a team of professionals.

- **First Deliverable and Consulting**

  Delivering a preliminary report on the findings with suggestions on how to remediate those issues and providing consultation.

- **Reassessment**

  Verifying the status of the issues and whether there are any other complications in the fixes applied.

- **Final Deliverable**

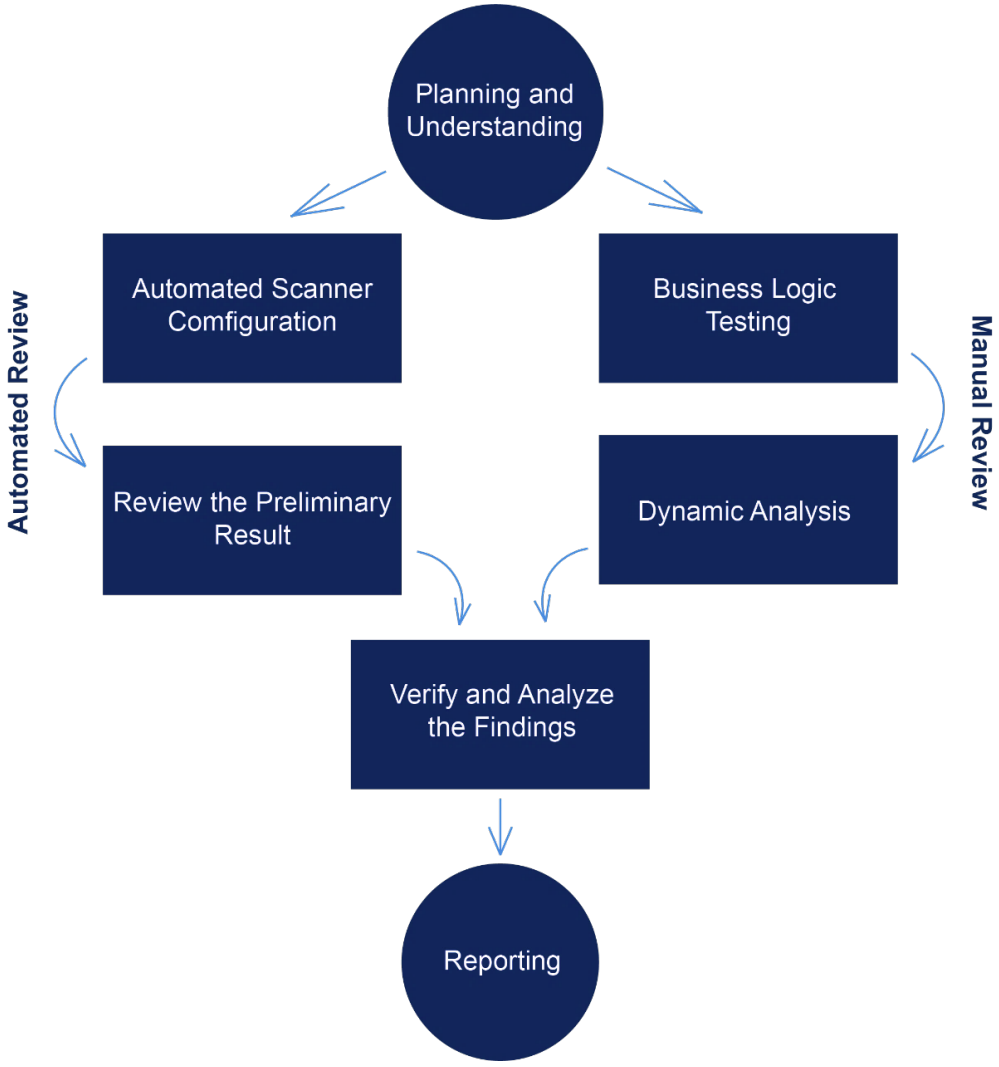
  Providing a full report with the detailed status of each issue.

The security audit process of CheckPoint includes three types testing:

1. Examining publicly available information about the token on social networks, including a detailed overview of the official website and analysis of the latest messages and opinions about the token.

2. Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

3. Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.



Remark: Manual and Automated review approaches can be mixed and matched including business logic analysis in terms of malicious doers' perspective

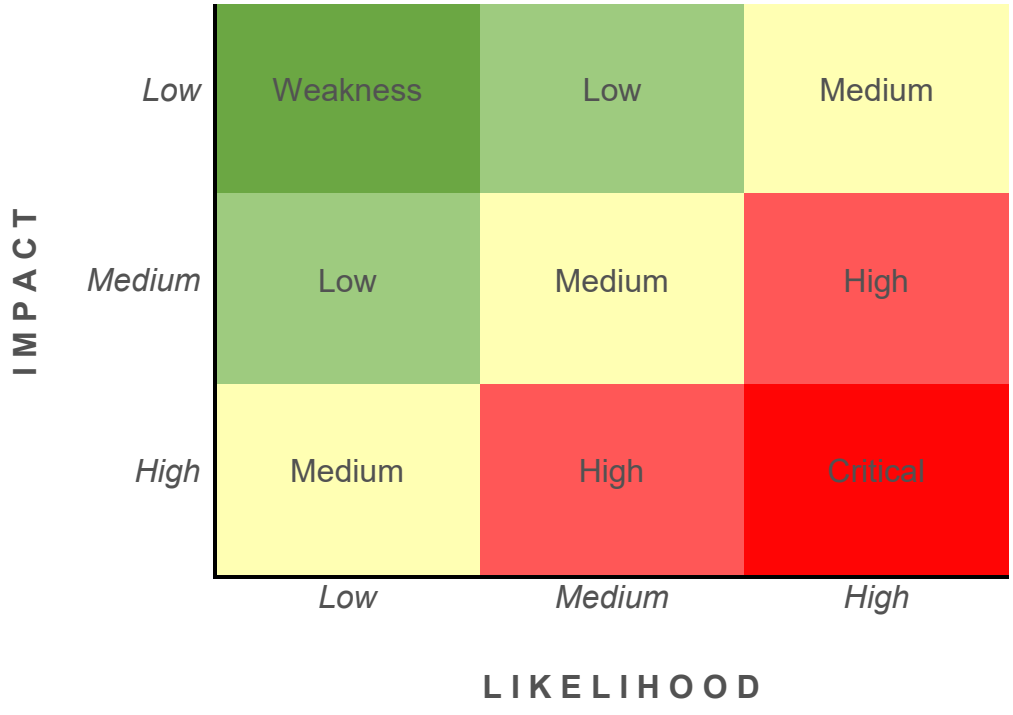
In particular, we perform the audit according to the following procedure:

- **Planning & Understanding**
  - determine scope of testing and understand application purpose and workflows;
  - identify key risk areas, including technical and business risks;
  - determine approach – which sections to review within the resource constraints and review method – automated, manual or mixed.
- **Automated Review**
  - adjust automated source code review tools to inspect the code for known unsafe coding patterns;
  - verify output of the tool in order to eliminate false positive result, and if necessary, adjust and re-run the code review tool.
- **Manual Review**
  - testing for business logic flaws requires thinking in unconventional methods;
  - identify unsafe coding behavior via static code analysis.
- **Reporting**
  - analyze the root cause of the flaws;
  - recommend coding process improvements.

3 Risk Level Classification

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology:

- **Likelihood** represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- **Impact** measures the technical loss and business damage of a successful attack.
- **Severity** demonstrates the overall criticality of the risk and calculated as the product of impact and likelihood values, illustrated in a twodimensional matrix. The shading of the matrix visualizes the different risk levels.



Remark: Likelihood and Impact are categorized into three levels: H, M, and L, i.e., High, Medium and Low respectively. Severity is determined by likelihood and impact and can be classified into five categories accordingly, i.e., Critical, High, Medium, Low and Weakness

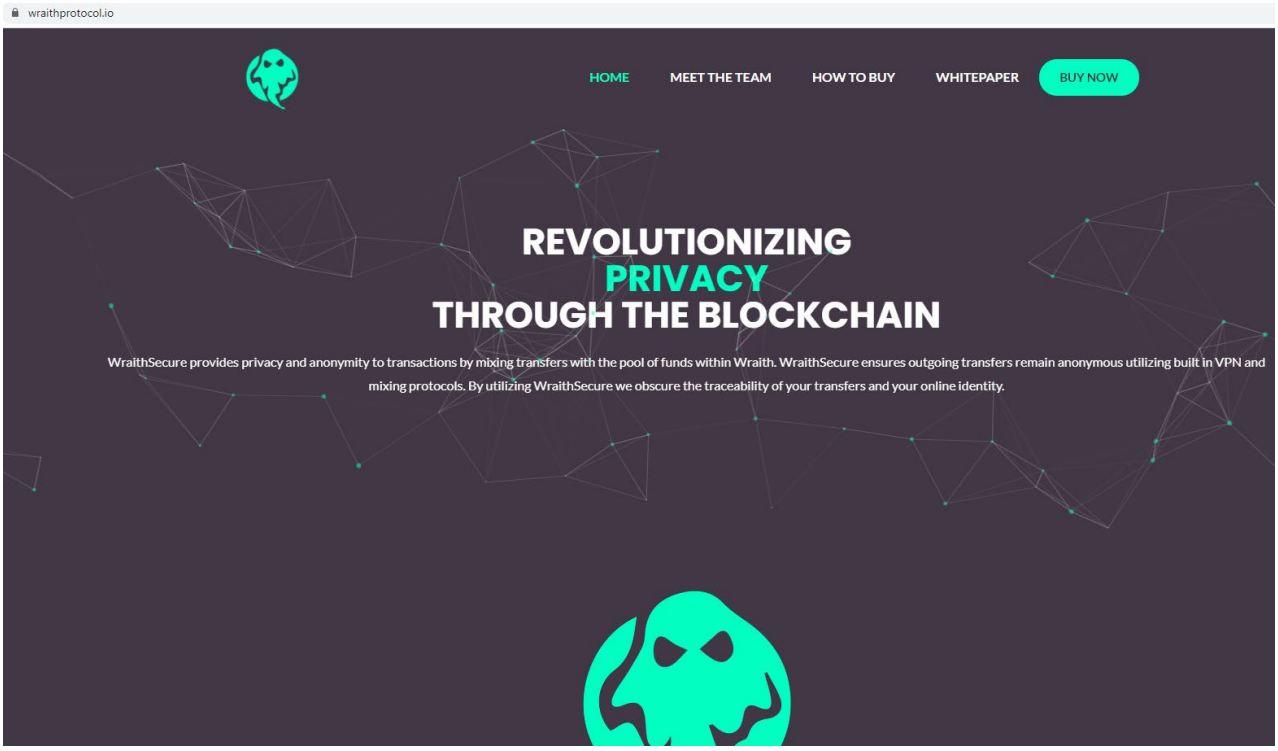
For prioritization of the vulnerabilities, we have adopted the scheme by five distinct levels for risk: Critical, High, Medium, Low, and Weakness. The risk level definitions are presented in table.

LEVEL	DESCRIPTION
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project

4 Project Overview

4.1 Communication Channels

https://wraithprotocol.io/



Website was registered on 21-07-2021, registration expires 21-07-2022.

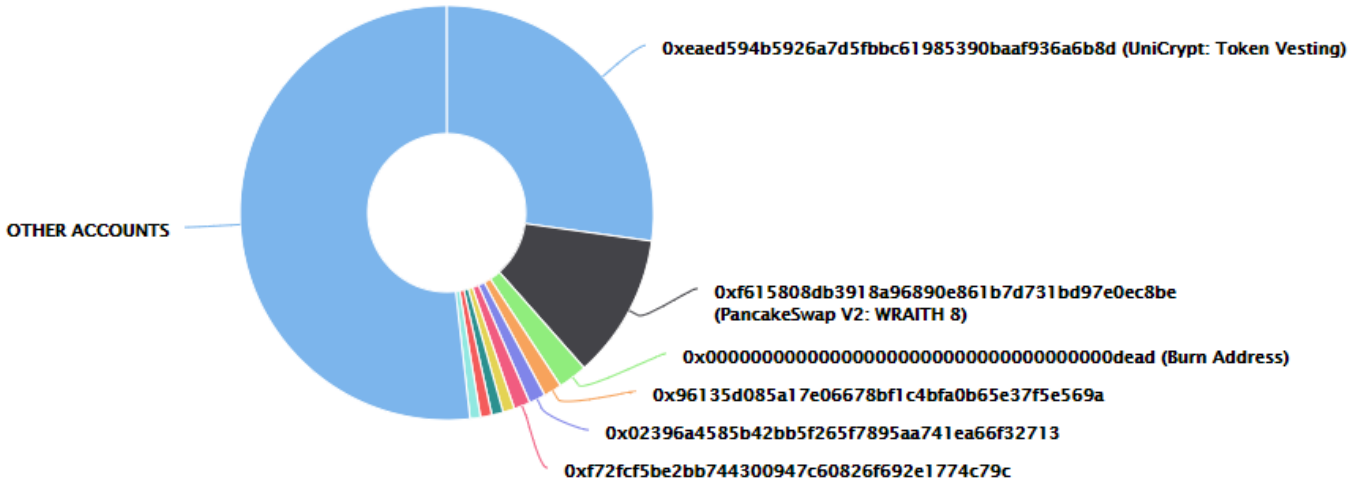
Above the image is an actual snapshot of the current live website of the project.

✓ Mobile Friendly	✓ 5 Social Media Networks
✓ No JavaScript Errors	✓ 4000+ Telegram Members
✓ Visionary Roadmap	✓ 1000+ Twitter Followers
✓ Spell Check	✓ Active voice chats
✓ Valid SSL Certificate	✓ No injected spam found
✓ Team Wallets	✓ No popus found

4.2 Smart Contract Details

Contract Name	Wraith
Contract Address	0x8b3b45E48bE6C31366ffd9dD4F29C1edFFcbA97D
Total Supply	10,000,000,000,000
Token Ticker	WRAITH
Decimals	9
Token Holders	2,495
Transactions Count	8,693
Top 100 Holders Dominance	75,51%
Liquidity Fee	5%
Marketing Fee	2%
Total Tax Fee	12%
Marketing Wallet Address	0x792ca25b3329351001a8e88c5fe97dad45425038
Wraith Charity Address	0x6adbe03ec04322322f5258c49f3508d33605bcbb
Contract Deployer Address	0x07453870eae19ef582a3c8a165a77cc1efb8eb65
Current Owner Address	0x07453870eae19ef582a3c8a165a77cc1efb8eb65

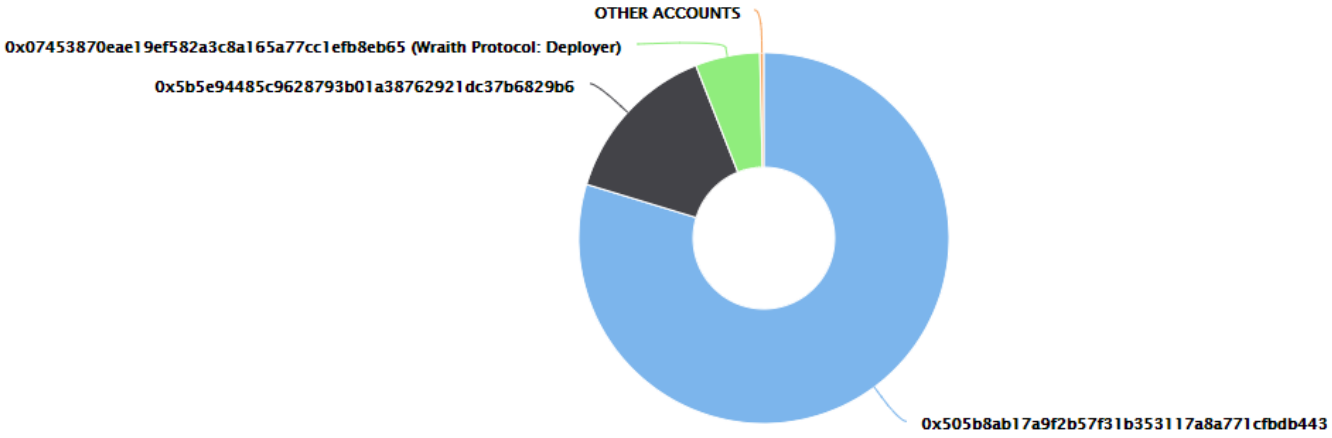
Wraith Top 10 Token Holders



Rank	Address	Quantity (Token)	Percentage
1	UniCrypt: Token Vesting	2,717,648,518,743.238279751	27.1765%
2	PancakeSwap V2: WRAITH 8	1,126,778,963,200.298269444	11.2678%
3	Burn Address	227,259,871,246.730082894	2.2726%
4	0x96135d085a17e06678bf1c4bfa0b65e37f5e569a	142,435,245,320.106904049	1.4244%
5	0x02396a4585b42bb5f265f7895aa741ea66f32713	129,627,460,679.999992656	1.2963%
6	0xf72fcf5be2bb744300947c60826f692e1774c79c	125,917,028,827.171453814	1.2592%
7	0x677e2c62afda432948c1cfebc2211b865ffc3540	90,820,538,340.09509058	0.9082%
8	0x908491353e82d72a7207cfeb08a0e2e70e897a4d	90,000,000,000	0.9000%
9	0x11a80a5192bd09f312f0e728f068283f5e6446cb	88,232,639,488.713989611	0.8823%
10	0xd878f9de0ac862ef2d0de5299348643ffd2a373d	81,102,199,204.399996848	0.8110%

[RISK] 27% tokens are locked

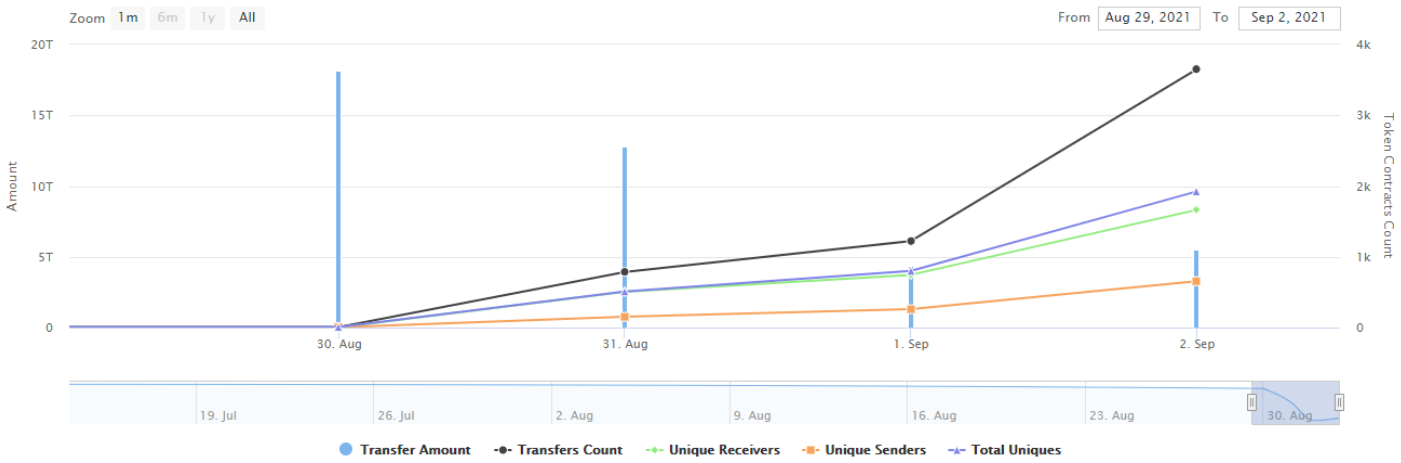
Baby Doge Coin Top 3 LP Token Holders



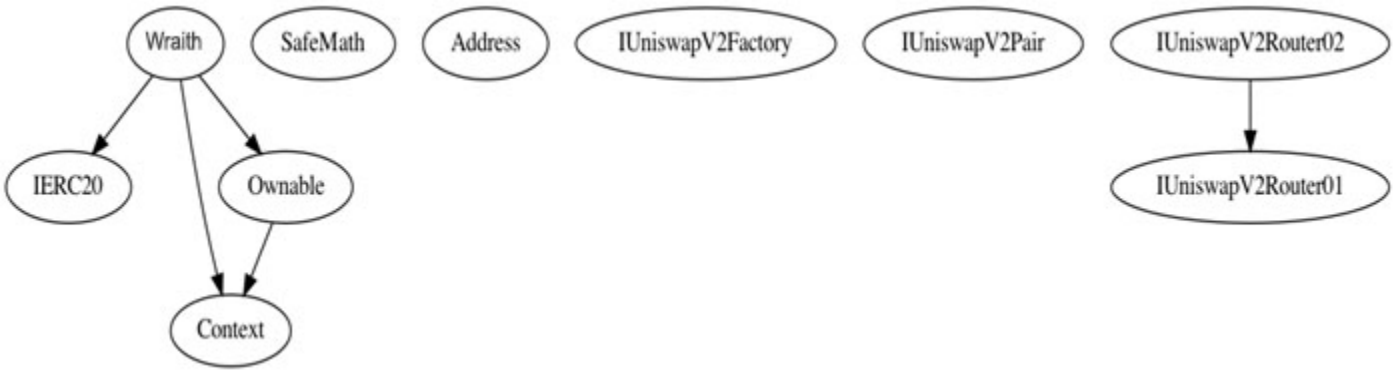
Rank	Address	Quantity (Token)	Percentage
1	0x505b8ab17a9f2b57f31b353117a8a771cfbdb443	751.29013254703612829 4	79.6817%
2	0x5b5e94485c9628793b01a38762921dc37b6829b6	135.211666186735732584	14.3405%
3	Wraith Protocol: Deployer	53.156491630564805506	5.6378%

[RISK] 2 wallet have 95% LP tokens

Baby Doge Coin Contract Interaction Details



4.3 Contract Function Details



$ = payable function
# = non-constant function
[Int] = Internal
[Pub] = Public
[Prv] = Private
[Ext] = External

+ [Int] IERC20
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] transfer #
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transferFrom #

+ [Lib] SafeMath
  - [Int] add
  - [Int] sub
  - [Int] sub
  - [Int] mul
  - [Int] div
  - [Int] div
  - [Int] mod
  - [Int] mod

+ [Lib] Address
  - [Int] isContract
  - [Int] sendValue #
  - [Int] functionCall #
  - [Int] functionCall #
  - [Int] functionCallWithValue #
  - [Int] functionCallWithValue #
  - [Prv] functionCallWithValue #

+ Ownable (Context)
  - [Pub] <Constructor> #
  - [Pub] owner
  - [Pub] renounceOwnership #
    - modifiers: onlyOwner
  - [Pub] transferOwnership #
    - modifiers: onlyOwner
  - [Pub] getUnlockTime
  - [Pub] getTime
  - [Pub] lock #
    - modifiers: onlyOwner
  - [Pub] unlock #

+ [Int] IUniswapV2Factory
  - [Ext] feeTo
  - [Ext] feeToSetter
  - [Ext] getPair
  - [Ext] allPairs
  - [Ext] allPairsLength
  - [Ext] createPair #
  - [Ext] setFeeTo #
  - [Ext] setFeeToSetter #

+ [Int] IUniswapV2Pair
  - [Ext] name
  - [Ext] symbol
  - [Ext] decimals
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transfer #
  - [Ext] transferFrom #
  - [Ext] DOMAIN_SEPARATOR
  - [Ext] PERMIT_TYPEHASH
  - [Ext] nonces
  - [Ext] permit #
  - [Ext] MINIMUM_LIQUIDITY
  - [Ext] factory
  - [Ext] token0
  - [Ext] token1
  - [Ext] getReserves
  - [Ext] price0CumulativeLast
  - [Ext] price1CumulativeLast
  - [Ext] kLast
  - [Ext] burn #
  - [Ext] swap #
  - [Ext] skim #
  - [Ext] sync #
  - [Ext] initialize #

+ [Int] IUniswapV2Router01
  - [Ext] factory
  - [Ext] WETH
  - [Ext] addLiquidity #
  - [Ext] addLiquidityETH $
  - [Ext] removeLiquidity #
  - [Ext] removeLiquidityETH #
  - [Ext] removeLiquidityWithPermit #
  - [Ext] removeLiquidityETHWithPermit #
  - [Ext] swapExactTokensForTokens #
  - [Ext] swapTokensForExactTokens #
  - [Ext] swapExactETHForTokens $
  - [Ext] swapTokensForExactETH #
  - [Ext] swapExactTokensForETH #
  - [Ext] swapETHForExactTokens $
  - [Ext] quote
  - [Ext] getAmountOut
  - [Ext] getAmountIn
  - [Ext] getAmountsOut
  - [Ext] getAmountsIn

+ [Int] IUniswapV2Router02 (IUniswapV2Router01)
  - [Ext] removeLiquidityETHSupportingFeeOnTransferTokens #
  - [Ext] removeLiquidityETHWithPermitSupportingFeeOnTransferTokens #
  - [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
  - [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens $
  - [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

+ WRAITH (Context, IERC20, Ownable)
  - [Pub] <Constructor> #
  - [Pub] name
  - [Pub] symbol
  - [Pub] decimals
  - [Pub] totalSupply
  - [Pub] balanceOf
  - [Pub] transfer #
  - [Pub] allowance
  - [Pub] approve #
  - [Pub] transferFrom #
  - [Pub] increaseAllowance #
  - [Pub] decreaseAllowance #
  - [Pub] minimumTokensBeforeSwapAmount
  - [Prv] deliver #
  - [Prv] reflectionFromToken
  - [Prv] tokenFromReflection
  - [Prv] _approve #
  - [Prv] _transfer #
  - [Prv] swapAndLiquify #
    - modifiers: onlyOwner

- [Prv] swapTokensForEth #
- [Prv] addLiquidity #
- [Prv] _tokenTransfer #
- [Prv] _transferStandard #
- [Prv] _transferToExcluded #
- [Prv] _transferFromExcluded #
- [Prv] _transferBothExcluded #
- [Prv] _getRate
- [Prv] _getValues
- [Prv] _getTValues
- [Prv] _getRValues
- [Prv] _getCurrentSupply
- [Prv] _takeLiquidity #
- [Prv] calculateFee
- [Prv] removeAllFee #
- [Prv] restoreAllFee #
- [Pub] isExcludedFromFee
- [Pub] excludeFromFee #
  - modifiers: onlyOwner
- [Pub] includeInFee #
  - modifiers: onlyOwner
- [Ext] setTaxes #
  - modifiers: onlyOwner
- [Ext] setMaxTxAmount #
  - modifiers: onlyOwner
- [Ext] setNumTokensBeforeSwap #
  - modifiers: onlyOwner
- [Ext] setMarketingWalletAddress #
  - modifiers: onlyOwner
- [Ext] setWraithCharityWalletAddress #
  - modifiers: onlyOwner
- [Pub] setSwapAndLiquifyEnabled #
  - modifiers: onlyOwner
- [Pub] setSwapAndLiquifyByLimitOnly #
  - modifiers: onlyOwner
- [Ext] prepareForPreSale #
  - modifiers: onlyOwner
- [Ext] prepareForLaunch #
  - modifiers: onlyOwner
- [Prv] transferToAddressETH #
- [Pub] changeRouterVersion #
  - modifiers: onlyOwner
- [Ext] <Fallback> $

4.4 Issues Checking Status

CHECKING ITEM	NOTES	RESULT
Arbitrary Jump with Function Type Variable	N / A	PASS
Arithmetic Accuracy Deviation	N / A	PASS
Assert Violation	N / A	PASS
Authorization through tx.origin	N / A	PASS
Business Logic	N / A	PASS
Code with No Effects	N / A	PASS
Critical Solidity Compiler	N / A	PASS
Delegatecall to Untrusted Callee	N / A	PASS
Design Logic	N / A	PASS
DoS with Block Gas Limit	N / A	LOW RISK
DoS with Failed Call	N / A	PASS
Function Default Visibility	N / A	PASS
Hash Collisions With MVLA	N / A	PASS
Incorrect Constructor Name	N / A	PASS
Incorrect Inheritance Order	N / A	PASS
Integer Overflows and Underflows	N / A	PASS
Lack of Proper Signature Verification	N / A	PASS
Message Call with Hardcoded Gas Amount	N / A	PASS
Missing Protection Against SRA	N / A	PASS
Presence of Unused Variables	N / A	PASS
Reentrancy	N / A	PASS

CHECKING ITEM	NOTES	RESULT
Requirement Violation	N / A	PASS
Right-To-Left-Override Control Character	N / A	PASS
Shadowing State Variables	N / A	PASS
Signature Malleability	N / A	PASS
State Variable Default Visibility	N / A	PASS
Timestamp Dependence	N / A	PASS
Transaction Order Dependence	N / A	PASS
Typographical Error	N / A	PASS
Unencrypted Private Data On-Chain	N / A	PASS
Unexpected Ether balance	N / A	PASS
Uninitialized Storage Pointer	N / A	PASS
Use of Deprecated Solidity Functions	N / A	PASS
Weak Sources of Randomness From CA	N / A	PASS
Write to Arbitrary Storage Location	N / A	PASS

Remark: To evaluate the risk, we go through a list of check items and each would be labeled with a severity category. For one check item, if our tool or analysis does not identify any issue, the contract is considered safe regarding the check item

4.5 Detailed Findings Information

[RISK] DoS with Block Gas Limit

- The function `_getCurrentSupply()` also uses the loop for evaluating total supply. It also could be aborted with out-of-gas exception if there will be a long excluded addresses list.

```
function _getCurrentSupply() private view returns(uint256, uint256) {
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
        rSupply = rSupply.sub(_rOwned[_excluded[i]]);
        tSupply = tSupply.sub(_tOwned[_excluded[i]]);
    }
    if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
    return (rSupply, tSupply);
}
```

Recommendation: Make sure the length of the excluded array is not too long

[RISK] Owner Privileges (in the period when the owner is not renounced)

- The owner of the contract can exclude/include accounts from/in transfer fees and reward distribution.

```
function excludeFromFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = true;
}

function includeInFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = false;
}
```

- The owner of the contract can exclude accounts from transfer fees and reward distribution.

```
function excludeFromFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = true;
}
```

- Owner can change fees.

```
function setTaxes(uint256 newBurnFee, uint256 newLiquidityTax, uint256 newMarketingTax, uint256 newWraithCharityTax) external onlyOwner() {
    _burnFee = newBurnFee;
    _liquidityFee = newLiquidityTax;
    _marketingFee = newMarketingTax;
    _wraithCharityFee = newWraithCharityTax;
    _totalTaxPercent = _burnFee.add(_liquidityFee).add(_marketingFee).add(_wraithCharityFee);
    _prevTotalTaxPercent = _totalTaxPercent;
}
```

- Owner can change minimum tokens before swap.

```
function setNumTokensBeforeSwap(uint256 newLimit) external onlyOwner() {
    minimumTokensBeforeSwap = newLimit;
}
```

- Owner can change Marketing and Wraith Charity wallets.

```
function setMarketingWalletAddress(address newAddress) external onlyOwner() {
    marketingWalletAddress = payable(newAddress);
}

function setWraithCharityWalletAddress(address newAddress) external onlyOwner() {
    wraithCharityWalletAddress = payable(newAddress);
}
```

- Owner can disable and enable SwapAndLiquifyByLimitOnly.

```
function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
    swapAndLiquifyEnabled = _enabled;
    emit SwapAndLiquifyEnabledUpdated(_enabled);
}

function setSwapAndLiquifyByLimitOnly(bool newValue) public onlyOwner {
    swapAndLiquifyByLimitOnly = newValue;
}
```

- Owner can enable prepareForPreSale and prepareForLaunch presets.

```
function prepareForPreSale() external onlyOwner {
    setSwapAndLiquifyEnabled(false);
    _totalTaxPercent = 0;
    _prevTotalTaxPercent = 0;
    _maxTxAmount = 1000000000 * 10**6 * 10**9;
}

function prepareForLaunch() external onlyOwner {
    setSwapAndLiquifyEnabled(true);
    _totalTaxPercent = _burnFee.add(_liquidityFee).add(_marketingFee).add(_wraithCharityFee);
    _prevTotalTaxPercent = _totalTaxPercent;
    _maxTxAmount = 3000000 * 10**6 * 10**9;
}
```

- Owner can change router address.

```
function changeRouterVersion(address newRouterAddress) public onlyOwner returns(address newPairAddress) {

    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(newRouterAddress);

    newPairAddress = IUniswapV2Factory(_uniswapV2Router.factory()).getPair(address(this), _uniswapV2Router.WETH());

    if(newPairAddress == address(0)) //Create If Doesnt exist
    {
        newPairAddress = IUniswapV2Factory(_uniswapV2Router.factory())
            .createPair(address(this), _uniswapV2Router.WETH());
    }

    uniswapV2Pair = newPairAddress; //Set new pair address
    uniswapV2Router = _uniswapV2Router; //Set new router address
}

 //to recieve ETH from uniswapV2Router when swaping
receive() external payable {}
```

- Owner can lock and unlock.

```
function lock(uint256 time) public virtual onlyOwner {
    _previousOwner = _owner;
    _owner = address(0);
    _lockTime = now + time;
    emit OwnershipTransferred(_owner, address(0));
}

//Unlocks the contract for owner when _lockTime is exceeds
function unlock() public virtual {
    require(_previousOwner == msg.sender, "You don't have permission to unlock");
    require(now > _lockTime , "Contract is locked until 7 days");
    emit OwnershipTransferred(_owner, _previousOwner);
    _owner = _previousOwner;
}
```

5 Audit Result

LEVEL	ISSUES
Weakness	**DoS with Block Gas Limit (1)**
Medium	**Owner Privilegies (9)**

1. The contract utilizes SafeMath libraries along with following the ERC20 standard.

2. The owner has the ability to set and update a maximum transaction percent at any time, which will impose a limit to the number of tokens that can be transferred during any given transaction. Maximum value not set. Maximum and minimum values not set. **[RISK]**

3. There is a 'liquidity fee', a ‘marketing fee’ and a ‘reflection fee’ on all transactions for any address that participates in a transfer. The owner has the ability to modify these fees from 10 to 100 percent, at any time. Maximum value not set. **[RISK]**

4. A portion of the tax fee is redistributed to existing token holders instantly and automatically at the time of each transaction.

5. The owner of the contract can exclude accounts from transfer fees and reward distribution.

6. The owner can sets the _maxTxAmount to 0, thus disallowing all non-whitelisted addresses to make any transfers.

7. The owner has the ability to use the ‘lock’. function in order to temporarily set ownership to address(0). Ownership is restored after the duration of time determined by the owner has passed and they use the 'unlock' function. Ownership can additionally be restored (even if ownership was previously renounced), by using the unlock function a second time.

8. Ownership has not been renounced.

5.1 Findings Summary

Wraith

Medium Risk Level

- ✓ **No external vulnerabilities were identified within the smart contract's code**
- ✓ **We strongly recommend that the team renounces ownership**
- ✓ **Please ensure trust in the team prior to investing as they have substantial control within the ecosystem**
- ✓ **We strongly recommend that the contract owners remove errors and re-audit**

6 Disclamer

CheckPoint team issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these. For the facts that occurred or existed after the issuance, CheckPoint is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to CheckPoint by the information provider till the date of the insurance report. CheckPoint is not responsible for the background and other conditions of the project.

This security audit is not produced to supplant any other type of assessment and does not guarantee the discovery of all security vulnerabilities within the scope of the assessment. However, we warrant that this audit is conducted with goodwill, professional approach, and competence. Since an assessment from one single party cannot be confirmed to cover all possible issues within the smart contract(s), CheckPoint suggests conducting multiple independent assessments to minimize the risks. Lastly, nothing contained in this audit report should be considered as investment advice.

CheckPoint

Website
https://checkpoint.report

E-mail
contact@checkpoint.report

Telegram
@checkpointreport

Github
https://github.com/checkpointreport