



CheckPoint

Token Security Audit Report Prepared for ForeverSmash

[v.1.0]

October 2021

Document Properties

Client	ForeverSmash
Platform	Binance Smart Chain
Language	Solidity
Codebase	0x496D6285db87a1934265321dE599eAce554Ad517

Audit Summary

Delivery Date	13.10.2021
Audit Methodology	Static Analysis, Manual Review
Auditor(s)	Tuomo Nieminen
Classification	Public
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	4
2 Audit Methodology	5
3 Risk Level Classification	8
4 Project Overview	10
4.1 Communication Channels	10
4.2 Smart Contract Details	12
4.3 Contract Function Details	14
4.4 Issues Checking Status	17
4.5 Detailed Findings Information	29
5 Audit Result	23
5.1 Findings Summary	24
6 Disclaimer	25

1 Executive Summary

On 13/10/2021, CheckPoint conducted a full audit for the ForeverSmash 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 were done 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.



ForeverSmash (SMASH) Medium Risk Level

Communication Channels

Website Content Analysis,
Social Media Listening

Smart Contract Code

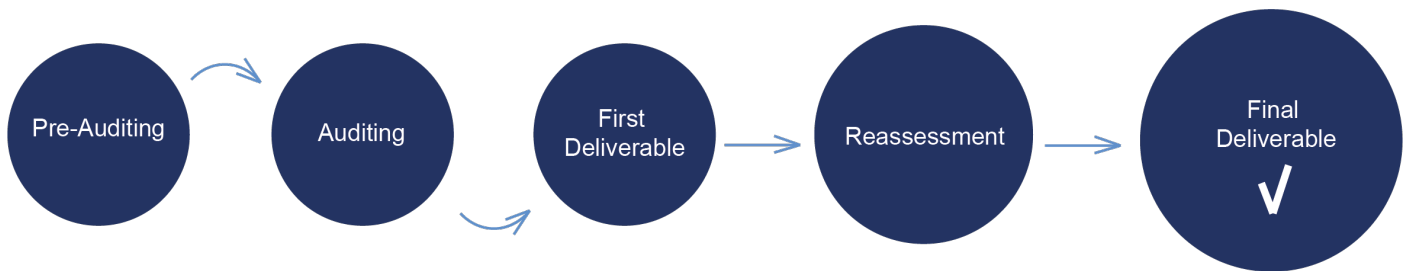
Smart Contract Details, Contract Function Details,
Issues Checking Status, Detailed Findings
Information



**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**

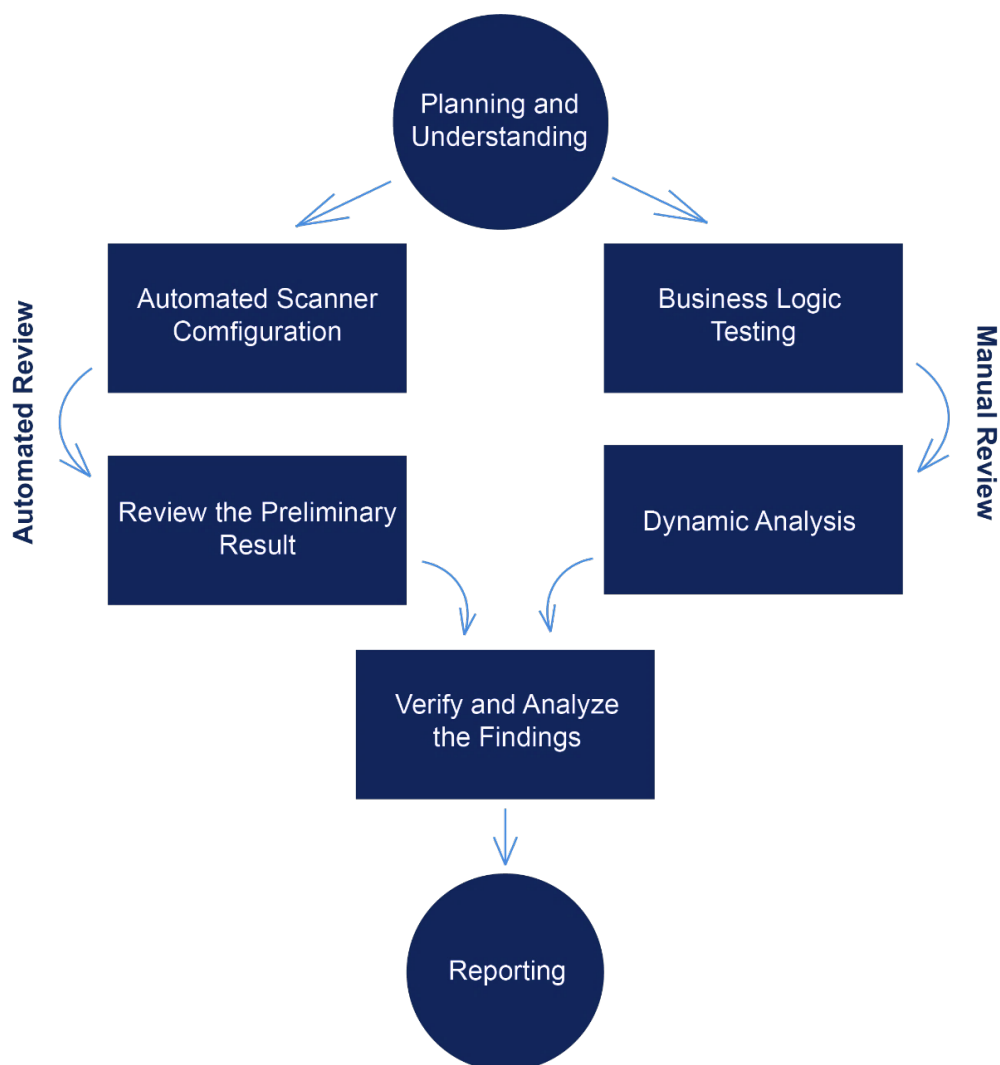
Verifying the status of the issues and whether there are any other complications in the fixes applied.

- **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.

IMPACT	Low	Weakness	Low	Medium
	Medium	Low	Medium	High
	High	Medium	High	Critical
		Low	Medium	High
		LIKELIHOOD		

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

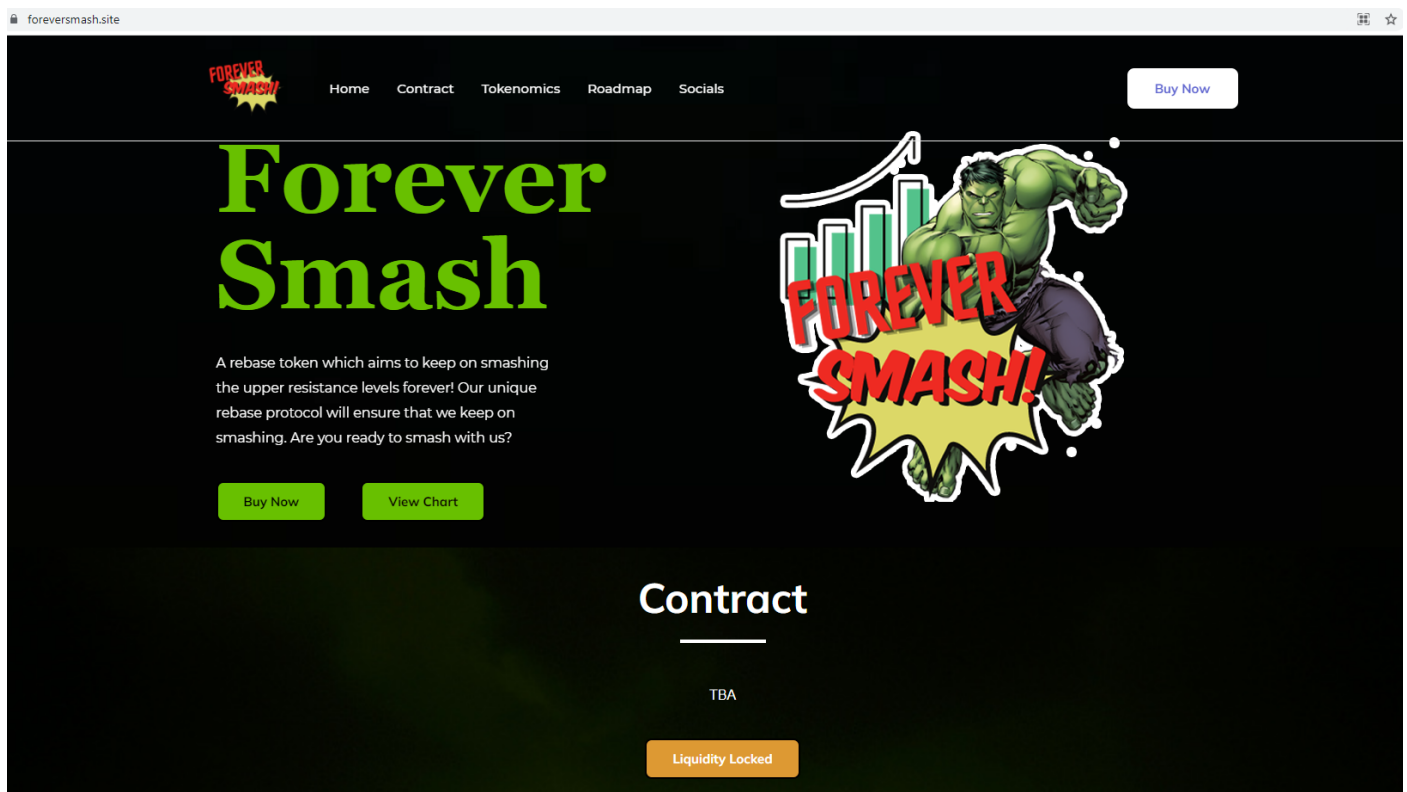
For prioritization of the vulnerabilities, we have adopted the scheme by five distinct levels for risk: Critical, High, Medium, Low, and Weakness. The risk level definitions are presented in table.

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://foreversmash.site/>



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

- | | |
|--------------------------------|---|
| ✓ Mobile Friendly | ✓ 2 Social Media Networks |
| ✓ No JavaScript Errors | ✓ <1000 Telegram Members [RISK] |
| ✓ Visionary Roadmap | ✓ <100 Twitter Followers [RISK] |
| ✓ Spell Check | ✓ No Active Voice Chats [RISK] |
| ✓ Valid SSL Certificate | ✓ No Injected Spam and Popus Found |



Remark: This page contains active links

4.2 Smart Contract Details

Contract Name ForeverSmash

Contract Address 0x496D6285db87a1934265321dE599eAce554Ad517

Total Supply 1,000,000,000,000,000

Token Ticker SMASH

Decimals 9

Token Holders 13

Transactions Count 86

Top 10 Holders Dominance 99,73%

Buyback Fee 4%

Ecosystem Fee 2%

Liquidity Fee 2%

Marketing Fee 4%

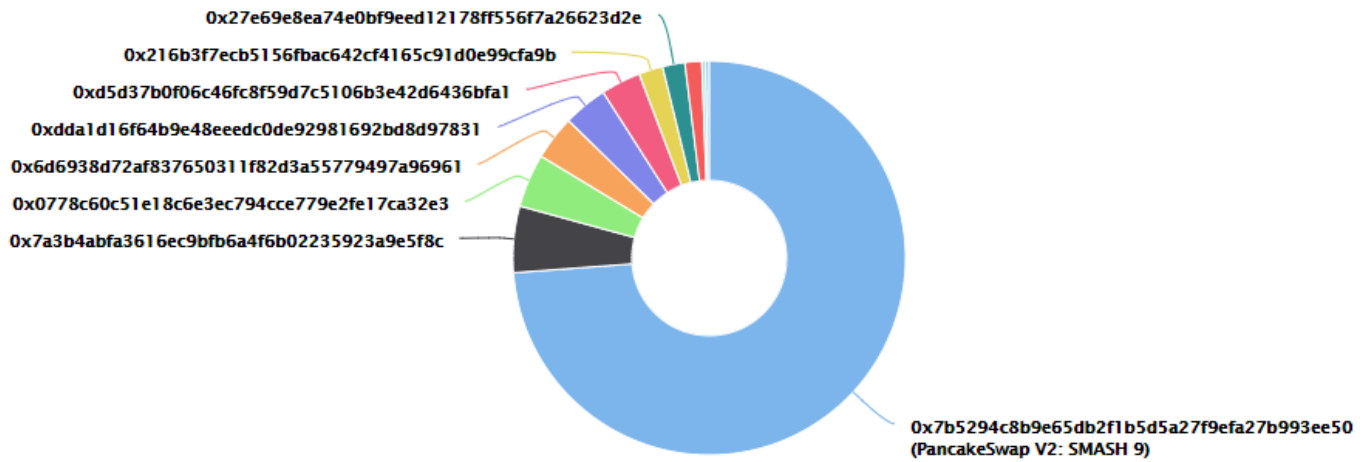
Fee Denominator 100


Pair Contract 0x7b5294c8b9e65db2f1b5d5a27f9efa27b993ee50

Contract Deployer Address 0xd6cb0e911a21d6a906487e08d899c8364b91e87a

Current Owner Address 0xd6cb0e911a21d6a906487e08d899c8364b91e87a

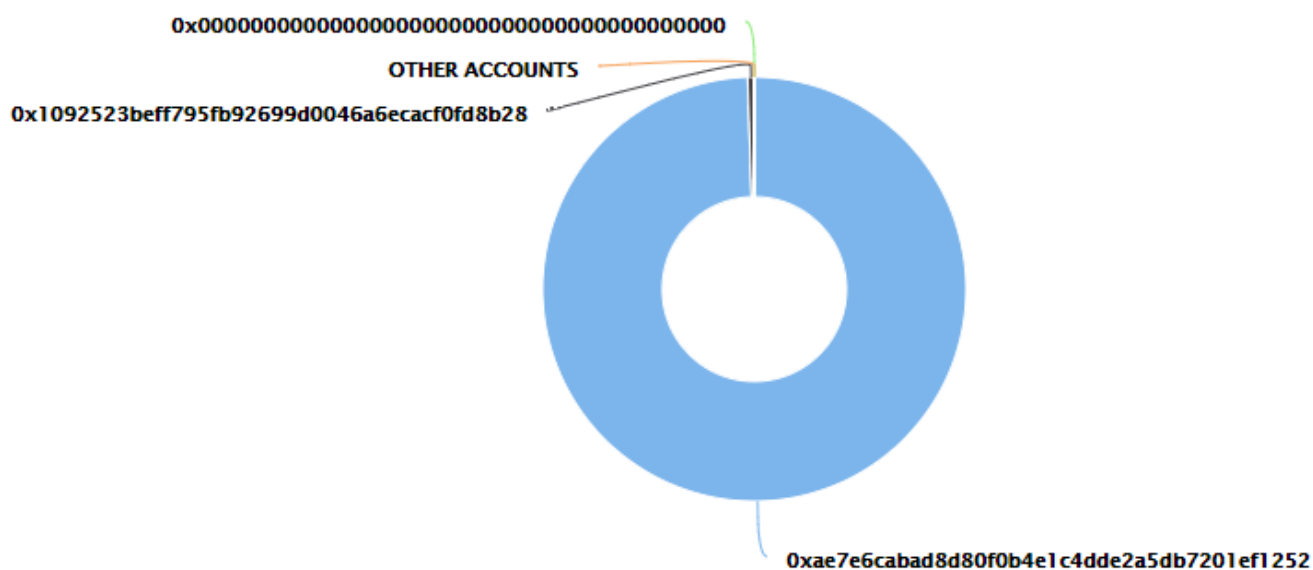
ForeverSmash Top 10 Token Holders




Rank	Address	Quantity (Token)	Percentage
1	 PancakeSwap V2: SMASH 9	738,128,342,960,263.869577174	73.8128%
2	0x7a3b4abfa3616ec9bfb6a4f6b02235923a9e5f8c	54,432,221,771,427.869574732	5.4432%
3	0x0778c60c51e18c6e3ec794cce779e2fe17ca32e3	44,000,000,000,000	4.4000%
4	0x6d6938d72af837650311f82d3a55779497a96961	36,433,143,436,206.746849742	3.6433%
5	0xdda1d16f64b9e48eedc0de92981692bd8d97831	36,252,040,551,574.190655794	3.6252%
6	0xd5d37b0f06c46fc8f59d7c5106b3e42d6436bfa1	32,760,816,000,000.088353013	3.2761%
7	0x216b3f7ecb5156fbac642cf4165c91d0e99cfa9b	19,896,510,436,558.37739629	1.9897%
8	0x27e69e8ea74e0bf9eed12178ff556f7a26623d2e	18,291,066,972,546.795728343	1.8291%
9	0x1a981151f2ef30f1bca94f0bdb0d4c2f4fc2d0e6	13,829,323,151,747.566727131	1.3829%
10	0x5e3773713f0a20806168d633652b905ee7253f83	3,285,918,510,844.628505092	0.3286%

✓ PancakeSwap holds ~74% of the token's supply as liquidity

ForeverSmash Top 3 LP Token Holders



Rank	Address	Quantity (Token)	Percentage
1	 0xae7e6cabad8d0f0b4e1c4dde2a5db7201ef1252	994.9999999999999999005	99.5000%
2	0x1092523beff795fb92699d0046a6ecacf0fd8b28	4.99999999999999995	0.5000%
3	0x00	0.00000000000000001	0.0000%

- ✓ 1 wallet have ~100% LP tokens [RISK]
- ✓ LP tokens unlock in 3 days [RISK]

4.3 Contract Function Details

\$ = payable function

= non-constant function

[Int] = Internal

[Pub] = Public

[Prv] = Private

[Ext] = External

+ [Lib] SafeMath

- [Int] add
- [Int] sub
- [Int] sub
- [Int] mul
- [Int] div
- [Int] div
- [Int] mod
- [Int] mod

+ [Int] IERC20

- [Ext] totalSupply
- [Ext] balanceOf
- [Ext] transfer #
- [Ext] allowance
- [Ext] approve #
- [Ext] transferFrom #

+ [Int] InterfaceLP

- [Ext] sync #

+ [Int] ERC20Detailed (IERC20)

- [Pub] name
- [Pub] symbol
- [Pub] decimals

+ [Lib] SafeMathInt

- [Int] mul
- [Int] div
- [Int] sub
- [Int] add
- [Int] abs

+ Ownable (Context)

- [Int] <Constructor> #
- [Pub] owner
- [Pub] isOwner
- [Pub] renounceOwnership #
 - modifiers: onlyOwner
- [Pub] transferOwnership #

- modifiers: onlyOwner
- + [Int] IDEXFactory
 - [Ext] createPair #
- + [Int] IDEXRouter
 - [Ext] factory
 - [Ext] WETH
 - [Ext] addLiquidity #
 - [Ext] addLiquidityETH \$
 - [Ext] removeLiquidity #
 - [Ext] removeLiquidityETH #
 - [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
 - [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens \$
 - [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #
- + ForeverSmash (ERC20Detailed, Ownable)
 - [Ext] rebase #
 - modifiers: onlyMaster
 - [Pub] #
 - modifiers: ERC20Detailed
 - [Ext] setMaster #
 - modifiers: onlyOwner
 - [Ext] setLP #
 - modifiers: onlyOwner
 - [Ext] totalSupply
 - [Ext] balanceOf
 - [Ext] transfer #
 - modifiers: validRecipient, initialDistributionLock
 - [Ext] allowance
 - [Ext] transferFrom #
 - modifiers: validRecipient
 - [Int] _transferFrom #
 - [Int] _basicTransfer #
 - [Int] takeFee #
 - [Int] swapBack \$
 - [Ext] approve #
 - modifiers: initialDistributionLock
 - [Ext] increaseAllowance #
 - modifiers: initialDistributionLock
 - [Ext] decreaseAllowance #
 - modifiers: initialDistributionLock
 - [Ext] setInitialDistributionFinished #
 - modifiers: onlyOwner
 - [Ext] enableTransfer #
 - modifiers: onlyOwner
 - [Ext] setFeeExempt #
 - modifiers: onlyOwner
 - [Ext] checkFeeExempt

- [Ext] setMaxWalletExempt #
 - modifiers: onlyOwner
- [Ext] checkMaxWalletExempt
- [Ext] setMaxWalletToken #
 - modifiers: onlyOwner
- [Ext] checkMaxWalletToken
- [Int] shouldTakeFee
- [Int] shouldSwapBack
- [Ext] setSwapBackSettings #
 - modifiers: onlyOwner
- [Ext] setTargetLiquidity #
 - modifiers: onlyOwner
- [Ext] isNotInSwap
- [Ext] checkSwapThreshold
- [Ext] manualSync #
- [Ext] setFees #
 - modifiers: onlyOwner
- [Ext] setFeeRecievers #
 - modifiers: onlyOwner
- [Pub] rescueToken #
 - modifiers: onlyOwner
- [Pub] clearStuckBalance #
 - modifiers: onlyOwner
- [Priv] transferToAddressETH
- [Pub] getCirculatingSupply
- [Ext] sendPresale #
 - modifiers: onlyOwner
- [Pub] getLiquidityBacking
- [Pub] isOverLiquified

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	PASS
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
Requirement Violation	N / A	PASS

CHECKING ITEM	NOTES	RESULT
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 `sendPresale` uses the loop to send the presale. It could be aborted with out-of-gas exception if there will be a long excluded addresses list. Including an account in the reward again may result in unexpected behavior.

```
ftrace | funcSig
function sendPresale(address[] calldata recipients, uint256[] calldata values)
    external
    onlyOwner
{
    for (uint256 i = 0; i < recipients.length; i++) {
        _transferFrom(msg.sender, recipients[i], values[i]);
    }
}
```

Recommendation: Consider removing the `sendPresale` function. If this is not desired, consider avoiding it, especially on accounts with a significant balance.

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

The contract contains the following privileged functions that are restricted by the `onlyOwner` and the `onlyMaster`.

- The owner of the contract can set the Master.

```
ftrace | funcSig
function setMaster(address _master) external onlyOwner {
    master = _master;
}
```

- The Master can rebase the contract.

```
ftrace | funcSig
function rebase(uint256 epochI, int256 supplyDeltaI)
    external
    onlyMaster
    returns (uint256)
{
    require(!inSwap, "Try again");
    if (supplyDeltaI == 0) {
        emit LogRebase(epochI, totalSupply);
        return totalSupply;
    }

    if (supplyDeltaI < 0) {
        totalSupply = totalSupply.sub(uint256(-supplyDeltaI));
    } else {
        totalSupply = totalSupply.add(uint256(supplyDeltaI));
    }

    if (totalSupply > MAX_SUPPLY) {
        totalSupply = MAX_SUPPLY;
    }

    gonsPerFragment = TOTAL_GONS.div(totalSupply);
    pairContract.sync();

    emit LogRebase(epochI, totalSupply);
    return totalSupply;
}
```

- The owner of the contract can set initial distribution finished.

```
ftrace | funcSig
function setInitialDistributionFinished() external onlyOwner {
    initialDistributionFinished = true;
}
```

- The owner of the contract can enable transfer.

```
ftrace | funcSig
function enableTransfer(address _addrI) external onlyOwner {
    allowTransfer[_addrI] = true;
}
```

- The owner of the contract can set a fee exempt.

```
ftrace | funcSig
function setFeeExempt(address _addrI) external onlyOwner {
    isFeeExempt[_addrI] = true;
}
```

- The owner of the contract can set a max wallet exemption and a max wallet token.

```

fttrace | funcSig
function setMaxWalletExempt(address _addr!) external onlyOwner {
    isMaxWalletExempt[_addr!] = true;
}

fttrace | funcSig
function checkMaxWalletExempt(address _addr!) external view returns (bool) {
    return isMaxWalletExempt[_addr!];
}

fttrace | funcSig
function setMaxWalletToken(uint256 _num!, uint256 _denom!)
    external
    onlyOwner
{
    gonMaxWallet = TOTAL_GONS.div(_denom!).mul(_num!);
}

```

- The owner of the contract can set swap back settings and target liquidity.

```

fttrace | funcSig
function setSwapBackSettings(
    bool _enabled!,
    uint256 _num!,
    uint256 _denom!
) external onlyOwner {
    swapEnabled = _enabled!;
    gonSwapThreshold = TOTAL_GONS.div(_denom!).mul(_num!);
}

fttrace | funcSig
function setTargetLiquidity(uint256 target!, uint256 accuracy!) external onlyOwner {
    targetLiquidity = target!;
    targetLiquidityDenominator = accuracy!;
}

```

- The owner of the contract can clear the stuck balance.

```

fttrace | funcSig
function clearStuckBalance(uint256 amountPercentage!, address adr!) external onlyOwner {
    uint256 amountETH = address(this).balance;
    payable(adr!).transfer(
        (amountETH * amountPercentage!) / 100
    );
}

```

- The owner of the contract can set the fees and the fee receivers.

```
ftrace | funcSig
function setFees(
    uint256 _ecosystemFee!,
    uint256 _liquidityFee!,
    uint256 _buyBackFee!,
    uint256 _marketingFee!,
    uint256 _feeDenominator!
) external onlyOwner {
    ecosystemFee = _ecosystemFee;
    liquidityFee = _liquidityFee;
    buyBackFee = _buyBackFee;
    marketingFee = _marketingFee;
    totalFee = ecosystemFee.add(liquidityFee).add(marketingFee).add(buyBackFee);
    feeDenominator = _feeDenominator;
    require(totalFee < feeDenominator / 4);
}

ftrace | funcSig
function setFeeReceivers(
    address _autoLiquidityReceiver!,
    address _ecosystemFeeReceiver!,
    address _marketingFeeReceiver!,
    address _buyBackFeeReceiver!
) external onlyOwner {
    autoLiquidityReceiver = _autoLiquidityReceiver;
    ecosystemFeeReceiver = _ecosystemFeeReceiver;
    marketingFeeReceiver = _marketingFeeReceiver;
    buyBackFeeReceiver = _buyBackFeeReceiver;
}
```

- The owner of the contract can send a presale.

```
ftrace | funcSig
function sendPresale(address[] calldata recipients!, uint256[] calldata values!)
    external
    onlyOwner
{
    for (uint256 i = 0; i < recipients!.length; i++) {
        _transferFrom(msg.sender, recipients![i], values![i]);
    }
}
```

5 Audit Result

LEVEL	ISSUES
Weakness	DoS with Block Gas Limit (1)
Medium	Owner Privileges (10)

1. The contract utilizes SafeMath libraries along with following the ERC20 standard.
2. There is an 'Ecosystem fee', a 'Marketing fee', a 'Liquidity fee', and a 'Buyback Fee' on all transactions for any non-excluded address that participates in a transfer. The owner can update the tax rates to any percentage at any time.
3. This might deter investors as they could be wary that these fees might one day be set to 100% to force transfers to go to the owner.
4. The owner can also exclude and include users from the fee mechanism.
5. Transfers of the token are initially disabled until the team indicates token distribution has been completed.
6. The rebase function properly calls sync() on the pair contract to prevent theft-of-liquidity attacks that have occurred with other rebase tokens. The owner can add/remove other liquidity pools to this list to sync them after rebases.
7. The owner has the ability to update the Master & LP addresses at any time; as well as some variables used in calculating the rebase.

5.1 Findings Summary



ForeverSmash

Medium Risk Level

✓ No external vulnerabilities were identified within the smart contract's code

✓ The code is fully customized

✓ As with any presale, ensure trust in the team prior to investing

✓ Ensure trust in the team as they have substantial control over the ecosystem and will control the charity/marketing wallets

✓ ForeverSmash token was audited, and no issues were found

6 Disclaimer

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](https://t.me/checkpointreport)

Github

<https://github.com/checkpointreport>