



Building the Futuristic **Blockchain Ecosystem**

# SECURITY AUDIT REPORT

USR

# TOKEN OVERVIEW

## Risk Findings

Severity	Found
● High	0
● Medium	1
● Low	2
● Informational	1

## Points to note

- Owner can renounce the ownership.
- Owner can transfer the ownership.
- Owner can exclude wallets from fees.
- Owner can enable trading.
- Owner can pause trading.
- Owner can enable/disable burn.
- Owner can set buy liquidity percentage not more than 5%.
- Owner can set buyburn percentage not more than 5%.
- Owner can set sell user foundation percentage not more than 5%.
- Owner can set sell percentage not more than 5%.
- Owner can set transfer liquidity percentage not more than 3%.
- Owner can set transfer burn percentage not more than 3%.
- Owner can blacklist wallets.
- Owner can set swapthreshold.
- Owner can withdraw contract's ETH.

# TABLE OF CONTENTS

02	Token Overview	
03	Table of Contents	
04	Overview	
05	Contract Details	
06	Audit Methodology	
07	Vulnerabilities Checklist	
08	Risk Classification	
09	Static Analysis	
10	Testnet Version	
11	Manual Review	
16	About Expelee	
17	Disclaimer	

# OVERVIEW

The Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

<b>Audit Result</b>	<b>Passed</b>
<b>KYC Verification</b>	-
<b>Audit Date</b>	<b>07 November 2023</b>

# CONTRACT DETAILS

**Token Address:** –

**Name:** USR

**Symbol:** USR

**Decimals:** 18

**Network:** Binance smart chain

**Token Type:** ERC20

**Owner:** –

**Deployer:** –

**Token Supply:** 4200000000000

**Checksum:** 0ff4376de461c7768aeb7a75b5a3flee

**Testnet version:**

The tests were performed using the contract deployed on the Binance smart chain Testnet, which can be found at the following address:

<https://testnet.bscscan.com/address/0xbfd18c0a03e41643f6d68ec943dcbd6a1dc1f960#code>

# AUDIT METHODOLOGY

## Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.

## Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch, that lead to scams and rugpulls.

## Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
- Back-doors
- Vulnerability
- Accuracy
- Readability

## Tools

- DE
- Open Zeppelin
- Code Analyzer
- Solidity Code
- Compiler
- Hardhat

# VULNERABILITY CHECKS

Design Logic	Passed
Compiler warnings	Passed
Private user data leaks	Passed
Timestamps dependence	Passed
Integer overflow and underflow	Passed
Race conditions & reentrancy. Cross-function race conditions	Passed
Possible delays in data delivery	Passed
Oracle calls	Passed
Front Running	Passed
DoS with Revert	Passed
DoS with block gas limit	Passed
Methods execution permissions	Passed
Economy model	Passed
Impact of the exchange rate on the logic	Passed
Malicious event log	Passed
Scoping and declarations	Passed
Uninitialized storage pointers	Passed
Arithmetic accuracy	Passed
Cross-function race conditions	Passed
Safe Zeppelin module	Passed

# RISK CLASSIFICATION

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and access control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time. We categorize these vulnerabilities by the following levels:

## High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

## Medium Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

## Low Risk

Issues on this level are minor details and warnings that can remain unfixed.

## Informational

Issues on this level are minor details and warnings that can remain unfixed.



```

INFO:Detectors:
USDToken.transfer(address,address,uint256).burnTax (USDTokenUpdated.sol#768) is a local variable never initialized
USDToken.transfer(address,address,uint256).userFundationPercent (USDTokenUpdated.sol#768) is a local variable never initialized
USDToken.constructor(address).unlissapaymaster (USDTokenUpdated.sol#378) is a local variable never initialized
USDToken.transfer(address,address,uint256).liquidityTax (USDTokenUpdated.sol#787) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
USDToken.addLiquidity(uint256,uint256) (USDTokenUpdated.sol#723-736) ignores return value by unlissapaymaster.addLiquidityETH(value: ethAmount)(address(this),tokenAmount,0,0,address(this),block.timestamp) (USDT
ethUpdated.sol#728-736)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
USDToken.transfer(address,uint256).owner (USDTokenUpdated.sol#607) shadows:
- Disable.owner() (USDTokenUpdated.sol#182-184) (function)
USDToken.sell(address,address) (USDTokenUpdated.sol#618) shadows:
- Disable.owner() (USDTokenUpdated.sol#182-184) (function)
USDToken.spendAllFees(address,address,uint256).owner (USDTokenUpdated.sol#488) shadows:
- Disable.owner() (USDTokenUpdated.sol#182-184) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
USDToken.setBurnerFundationPercent(uint256) (USDTokenUpdated.sol#886-889) should emit an event for:
- _buyTaxPercent = userFundationPercent (USDTokenUpdated.sol#886)
USDToken.setLiquidityTaxPercent(uint256) (USDTokenUpdated.sol#811-813) should emit an event for:
- _buyLiquidityTaxPercent = _liquidityPercent (USDTokenUpdated.sol#813)
USDToken.setBurnerBurnPercent(uint256) (USDTokenUpdated.sol#818-821) should emit an event for:
- _burnTaxPercent = _burnPercent (USDTokenUpdated.sol#820)
USDToken.setSellUserFundationPercent(uint256) (USDTokenUpdated.sol#823-825) should emit an event for:
- sellTaxPercent = userFundationPercent (USDTokenUpdated.sol#823)
USDToken.setSellLiquidityPercent(uint256) (USDTokenUpdated.sol#838-839) should emit an event for:
- sellLiquidityPercent = _liquidityPercent (USDTokenUpdated.sol#838)
USDToken.setSellBurnPercent(uint256) (USDTokenUpdated.sol#839-848) should emit an event for:
- sellBurnTaxPercent = _burnPercent (USDTokenUpdated.sol#839)
USDToken.setTransferLiquidityPercent(uint256) (USDTokenUpdated.sol#862-867) should emit an event for:
- transferLiquidityTaxPercent = _liquidityPercent (USDTokenUpdated.sol#866)
USDToken.setTransferBurnPercent(uint256) (USDTokenUpdated.sol#868-872) should emit an event for:
- transferBurnTaxPercent = _burnPercent (USDTokenUpdated.sol#868)
USDToken.setTransferFundationPercent(uint256) (USDTokenUpdated.sol#882-884) should emit an event for:
- _sellTaxPercent = _sellTaxPercent (USDTokenUpdated.sol#883)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

```

# TESTNET VERSION

## 1- Enable Tarding (**passed**):

<https://testnet.bscscan.com/tx/0xd6b337ff0e860b46813cf4f0e2274b86d335d541a9b7fedae3d98b25e4369802>

## 2- Approve (**passed**):

<https://testnet.bscscan.com/tx/0xd6b337ff0e860b46813cf4f0e2274b86d335d541a9b7fedae3d98b25e4369802>

## 3- Set Buy Burn Percentage (**passed**):

<https://testnet.bscscan.com/tx/0x1bb410d1fc6f49b0983aceecf5724b5bde23e98b0db44273feb6bff3de77af79>

## 4- Set Buy Liquidity Percentage (**passed**):

<https://testnet.bscscan.com/tx/0xc00da1ef6f8409a4a22ca79b49881dedeeec1ec35e649ece3bdb336101bc33d>

## 5- Set Buy User Foundation Percentage (**passed**):

<https://testnet.bscscan.com/tx/0xc5c27dcd4670f4f53695750cf5a8b3e8f4fe7d0f5c7ceae401ef3c321005a32b>

## 6- Set Excluded from Fee (**passed**):

<https://testnet.bscscan.com/tx/0x0809f1cae2c49acbc26949c1074a41ed107721650de989bcbb42b35786c172d3>

## 7- Transfer Ownership(**passed**):

<https://testnet.bscscan.com/tx/0xfd8261dd2d37f15358886e0c33a60a61c507293776a09d46f85a043f29a62e03>

# MANUAL REVIEW

## Severity Criteria

Expelee assesses the severity of disclosed vulnerabilities according to methodology based on OWASP standarts.

Vulnerabilities are dividend into three primary risk categroies:

High

Medium

Low

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious input handling
- Escalation of privileges
- Arithmetic
- Gas use

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

# MEDIUM RISK FINDING

**Centralization** – Owner can blacklist wallets.

Severity: **Medium**

function: addToBlacklist

Status: Open

## Overview:

The owner can blacklist wallets from transferring of tokens for an indefinite period of time which is not recommended. Which can lock user's token.

```
function addToBlacklist(address account) public onlyOwner{
    require(!blacklisted[account], "Account is already
blacklisted");
    require(_msgSender() != account, "Cannot blacklist self");
    blacklisted[account] = true.
}
```

## Suggestion:

There should be a locking period so that the wallet cannot be locked for an indefinite Period of time.

# LOW RISK FINDING

**Severity:** Low

**subject:** Missing Events

**Status:** Open

**Overview:**

They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function setExcludedFromFee(address account, bool excluded) external  
onlyOwner {  
    _isExcludedFromFee[account] = excluded;  
}
```

```
function enableTrade() public onlyOwner  
    trade_open = true;  
}
```

```
function pauseTrade() public onlyOwner {  
    trade_open = false;  
}
```

```
function disableBurn(bool _status) public onlyOwner {  
    burn_disable = _status;  
}
```

**Suggestion:**

Add an event to these important functions where address updation is happening. This can also be marked as indexed event for better off-chain tracking.



# LOW RISK FINDING

**Severity:** Low

**subject:** Local variable shadowing

**Status:** Open

## Overview:

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice to avoid them.

```
function transfer(  
    address to,  
    uint256 amount  
) public virtual returns (bool) {  
    address owner = msg.sender;  
    _transfer(owner, to, amount);  
    return true;  
}
```

## Suggestion:

To reduce high gas fees. It is suggested to remove. unused code from the contract.

# LOW RISK FINDING

**Severity:** Information

**Subject:** remove virtual if the function is not going to be overridden

**Status:** Open

## Overview:

```
function _checkOwner() internal view virtual {
    require(owner() == _msgSender(), "Ownable: caller is not the owner");
}

/**
 * @dev Leaves the contract without owner. It will not be possible to call
 * `onlyOwner` functions anymore. Can only be called by the current owner.
 *
 * NOTE: Renouncing ownership will leave the contract without an owner,
 * thereby removing any functionality that is only available to the owner.
 */
function renounceOwnership() public virtual onlyOwner {
    _transferOwnership(address(0));
}

/**
 * @dev Transfers ownership of the contract to a new account (`newOwner`).
 * Can only be called by the current owner.
 */
function transferOwnership(address newOwner) public virtual onlyOwner { ///@audit
remove virtual if the function is not going to be overridden -- info
    require(
        newOwner != address(0),
        "Ownable: new owner is the zero address"
    );
    _transferOwnership(newOwner);
}
```

## Suggestion:

It is recommended to remove virtual if the function is not going to be overridden.

# ABOUT EXPELEE

Expelee is a product-based aspirational Web3 start-up. Coping up with numerous solutions for blockchain security and constructing a Web3 ecosystem from deal making platform to developer hosting open platform, while also developing our own commercial and sustainable blockchain.

 [www.expelee.com](http://www.expelee.com)



expeleeofficial



expelee



Expelee



expelee



expelee\_official



expelee-co

# expelee

Building the Futuristic **Blockchain Ecosystem**



# DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantess against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always do your own research and project yourselves from being scammed. The Expelee team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools.

Under no circumstances did Expelee receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Alway do your own research and protect yourselves from scams.

This document should not be presented as a reason to buy or not buy any particular token. The Expelee team disclaims any liability for the resulting losses.

The logo for Expelee, featuring the word "expelee" in a stylized font. The "ex" is in white, and "pelee" is in orange. The letters are bold and modern.

Building the Futuristic **Blockchain Ecosystem**