

# expelee

Building the Futuristic **Blockchain Ecosystem**

## Audit Report FOR






## Gremlins Coin

# OVERVIEW

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 :

 Audit Result	Passed
 KYC Verification	Done
 Audit Date	11 Sep 2022

*Be aware that smart contracts deployed on the blockchain aren't resistant to internal exploit, external vulnerability, or hack. For a detailed understanding of risk severity, source code vulnerability, functional hack, and audit disclaimer, kindly refer to the audit.*

**- Team Expelee**



# PROJECT DESCRIPTION

## Gremlins Coin

Gremlins Coin is a community-driven project focusing on providing value through a growing feature system.

Supply grows scarcer over time, offers passive dividend income, rewards a high-value asset, 24/7 support, expanding team, international relations, and constantly growing.

 [gremlinscoin.com](https://gremlinscoin.com)

 GremlinsCoin

 GremlinsCoin

*It's always good to check the social profiles of the project,  
before making your investment.*

**- Team Expelee**

# CONTRACT DETAILS

Contract Name

**Gremlins**

---

Optimization

**Yes with 200 runs**

---

Contract Address

**0x125161Cb17B1d30f684906F813590C3f7cEE2370**

---

Network

**BSC**

---

Language

**Solidity**

---

Total Supply

**1,000,000,000,000**

---

Decimals

**18**

---

Compiler

**v0.8.16+commit.07a7930e**

---

License

**MIT license**

---



# 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

# FUNCTION OVERVIEW

Can Take Back Ownership	Not Detected
Owner Change Balance	Not Detected
Blacklist	Detected
Modify Fees	Detected
Proxy	Not Detected
Whitelisted	Not Detected
Anti Whale	Detected
Trading Cooldown	Detected
Transfer Pausable	Not Detected
Cannot Sell All	Not Detected
Hidden Owner	Detected
Creator Address	0xBe1dc97f489B389Bca45626f9597088277f6d8d7
Creator Balance	1,000,000,000,000 Grem
Owner Address	0x00
Mint	Not Detected



# VULNERABILITY CHECKLIST

Design Logic	Passed
Compiler warnings.	Passed
Private user data leaks	Passed
Timestamp 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
Fallback function security	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 warning that can remain unfixed.

## Informational

---

Information level is to offer suggestions for improvement of efficacy or security for features with a risk free factor.



# MANUAL AUDIT

## Fast Overview

---

- taxes can be up to 25% on buy and 25% on sell
- Owner ship of contract is renounced to address zero, but contract is not handled by address zero (i.e previous owner has still control over it)
- Owner is able to set a limit for buying/selling/transferring/holding
- amountOwner is able to blacklist an arbitrary address

## Centralization Issues

---

- **High** - Owner is able to take Pancake LP tokens from contract using **ReleaseLP()** function after passing **\_liquidityUnlockTime**, this **Pancake LP** tokens can be used to remove tokens from pool

**Recommendation:**

remove this function or give explanation about why you have to get Pancake LP tokens from contract

**Gremlins Dev:**

*'the contract embed a LP generator and lock (as if it was done on a third party, the release LP function is just here to be able to release the LP after the Lock time as it would be on a third party vault.'*

- **High** - Owner must use CreateLP function to add liquidity into liquidity pool, after adding liquidity there will be a block range (max 40 block) in which all buyers get blacklisted.

**Recommendation:**

Block creation time is currently around 3 seconds, this means if using max block range (40 block) every buyer in first 2 minutes will get blacklisted which is too much comparing to sniper bots actiontime (5-10 seconds)

**Gremlins Dev:**

*"unfortunately bots have become smarter lately, and i saw them behaving and sniping up to 1 min after project launched (LP gen) i understand the raising of this point but having an antibot of a few seconds is almost useless nowadays"*

- **Medium** - Owner is able to set taxes up to 25% using **SetTaxes** function, this means 50% tax on total buy + sell

- **Medium** - Owner is able to set a gap between sells, this means that a wallet can not sell without passing that gap between its sells, this gap can be up to one day and is defined as MinTime in contract



```
function SetDelay (bool delayoption, uint256 mintime) external
onlySharedOwners {
    require(mintime <= 28800, "MinTime Can't be more than a Day" );
    MinTime = mintime;
    DelayOption = delayoption;
}
```

## Logical Issues

- **Medium** - `if(contractTokenBalance >= MaxTokenToSwap){contractTokenBalance = MaxTokenToSwap;}` doing this can cause some of tokens to be stuck inside contract

### Recommendation

:

```
delete contractTokenBalance = MaxTokenToSwap
```

### Gremlins Dev:

"contract token balance is used to limit the contract from selling too much tax tokens, the remaining tokens will be used at another swap and liquify over time"

- **Low** - at **SetLimits** function, `_swapmax` is **not** multiplied by `10 ** 18` at first condition, consider multiplying this condition as well

### Gremlins Dev:

"I'm using a temporary `MaxToken` variable to have in get functions a readable value, then multiplying is done afterward"

## Gas Optimizations

- at **SetTaxes**, this check is redundant

```
require(newMarketingTax >= 0 && newTeamTax >= 0 && newMultiUseTax >= 0 &&
0, "No tax can be negative");
```

```
newBuyTax >= 0 && newRewardTax >= 0 && newLiquidityTax >= 0 && newBurnTax >=
```

because all of this variables are uint, there is no negative

- number this line should be at start of `_transfer` function

```
if(amount == 0) {return;}
```

- always do multiply prior to dividing

## Suggestions:

- typo at this function : **RemoveharedOwner** => **RemoveSharedOwner**
- at **swapAndLiquify** function, dividing is done prior to multiplication, always try to do multiply prior to dividing **swapTokensForTOKEN1**, **swapTokensForBNB**
- emit an event from this functions:  
**AddSharedOwner**, **RemoveharedOwner**, **setProjectWallet**, **SetDelay**, **SetLimits**, **SetTaxes**, **CreateLP**, **ReleaseLP**, **SetJeetsTax**



## 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 guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always Do your own research and protect 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. Always 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.