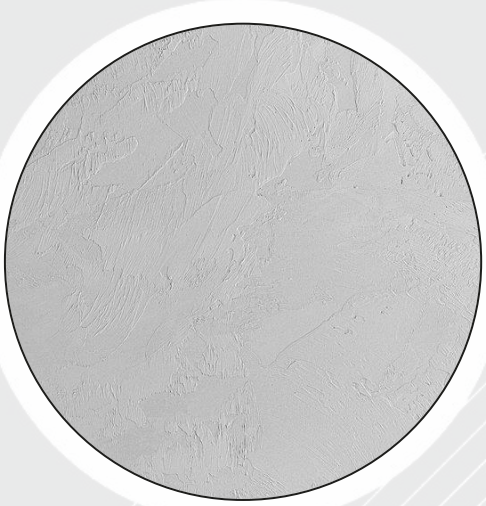
SPYWOLF

Security Audit Report



Audit prepared for

Gelato (Locker)

Completed on

October 24, 2024

OVERVIEW

This goal of this report is to review the main aspects of the project to help investors make an informative decision during their research process.

You will find a a summarized review of the following key points:

- ✔ Contract's source code
- ✔ Owners' wallets
- ✔ Tokenomics
- ✔ Team transparency and goals
- ✔ Website's age, code, security and UX
- ✔ Whitepaper and roadmap
- ✔ Social media & online presence

The results of this audit are purely based on the team’s evaluation and does not guarantee nor reflect the projects outcome and goal

- SPYWOLF Team -

TABLE OF CONTENTS

Gelato (Locker)

PROJECT DESCRIPTION:

No info yet.

Release Date: Unavailable

Launchpad: Unavailable

Category: Locker

KEY RESULTS

Cannot mint new tokens	*
Cannot pause trading (honeypot)	*
Cannot blacklist an address	*
Cannot raise taxes over 25%?	*
No proxy contract detected	PASSED
Not required to enable trading	*
No hidden ownership	PASSED
Cannot change the router	PASSED
No cooldown feature found	*
Bot protection delay is lower than 5 blocks	*
Cannot set max tx amount below 0.05% of total supply	*
The contract cannot be self-destructed by owner	PASSED

For a more detailed and thorough examination of the heightened risks, refer to the subsequent parts of the report.

N/A = Not applicable for this type of contract

*Not applicable for this type of contract

CONTRACT INFO

Token Name
N/A

Symbol
N/A

Contract Address
unavailable

Network
unavailable

Language
Solidity

Deployment Date
unavailable

Contract Type
Token locker

Total Supply
N/A

Decimals
N/A

TAXES

Buy Tax
-%

Sell Tax
-%

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

SMART CONTRACT STATS

Calls Count	unavailable
External calls	unavailable
Internal calls	unavailable
Transactions count	unavailable
Last transaction time	unavailable
Deployment Date	unavailable
Create TX	unavailable
Owner	unavailable
Deployer	unavailable

TOKEN TRANSFERS STATS

Transfer Count	unavailable
Total Amount	unavailable
Median Transfer Amount	unavailable
Average Transfer Amount	unavailable
First transfer date	unavailable
Last transfer date	unavailable
Days token transferred	unavailable

FEATURED WALLETS

Owner address	unavailable
Marketing fee receiver	unavailable
LP address	unavailable

TOP 3 UNLOCKED WALLETS

unavailable	
unavailable	
unavailable	

VULNERABILITY ANALYSIS

ID	Title	
SWC-100	Function Default Visibility	**Passed**
SWC-101	Integer Overflow and Underflow	**Passed**
SWC-102	Outdated Compiler Version	**Passed**
SWC-103	Floating Pragma	**Passed**
SWC-104	Unchecked Call Return Value	**Passed**
SWC-105	Unprotected Ether Withdrawal	**Passed**
SWC-106	Unprotected SELFDESTRUCT Instruction	**Passed**
SWC-107	Reentrancy	**Passed**
SWC-108	State Variable Default Visibility	**Passed**
SWC-109	Uninitialized Storage Pointer	**Passed**
SWC-110	Assert Violation	**Passed**
SWC-111	Use of Deprecated Solidity Functions	**Passed**
SWC-112	Delegatecall to Untrusted Callee	**Passed**
SWC-113	DoS with Failed Call	**Passed**
SWC-114	Transaction Order Dependence	**Passed**
SWC-115	Authorization through tx.origin	**Passed**
SWC-116	Block values as a proxy for time	**Passed**
SWC-117	Signature Malleability	**Passed**
SWC-118	Incorrect Constructor Name	**Passed**

VULNERABILITY ANALYSIS

ID	Title	
SWC-119	Shadowing State Variables	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-122	Lack of Proper Signature Verification	Passed
SWC-123	Requirement Violation	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-125	Incorrect Inheritance Order	Passed
SWC-126	Insufficient Gas Griefing	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-129	Typographical Error	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed
SWC-131	Presence of unused variables	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Passed
SWC-134	Message call with hardcoded gas amount	Passed
SWC-135	Code With No Effects	Passed
SWC-136	Unencrypted Private Data On-Chain	Passed

VULNERABILITY ANALYSIS

NO ERRORS FOUND

MANUAL
CODE REVIEW

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 4 different threat levels.

THREAT 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.

FOUND THREATS

⚠ High Risk: 0

No high risk-level threats found in this contract.

⚠ Medium Risk: 0

No medium risk-level threats found in this contract.

⚠ Low Risk: 0

No low risk-level threats found in this contract.

FOUND THREATS

Informational

Owner can retrieve LP tokens of GELATO/WPLS only after lockTime is expired.

```
function endLock() public onlyOwner {
    require(block.timestamp >= lockTime, "LP tokens are still locked.");

    IDEXPair lpToken1 = IDEXPair(IDEXFactory(pulseRouterV1.factory()).getPair(pulseRouterV1.WPLS(), address(gelatoToken)));
    IDEXPair lpToken2 = IDEXPair(IDEXFactory(pulseRouterV2.factory()).getPair(pulseRouterV2.WPLS(), address(gelatoToken)));
    IDEXPair lpToken3 = IDEXPair(IDEXFactory(nineinchRouter.factory()).getPair(nineinchRouter.WPLS(), address(gelatoToken)));

    lpToken1.transfer(owner(), lpToken1.balanceOf(address(this)));
    lpToken2.transfer(owner(), lpToken2.balanceOf(address(this)));
    lpToken3.transfer(owner(), lpToken3.balanceOf(address(this)));

    lockEnded = true;
    emit LockEnded();
}
```

Owner can extend current lock time with up to 1 year per transaction. Initial lock time is set withing the constructor during contract's deployment.

```
function extendLockTime(uint256 _extraLockTime) public onlyOwner {
    require(_extraLockTime > 0 && _extraLockTime <= 31536000, "Lock time can only be extended up to one year per call.");

    lockTime += _extraLockTime;
    emit ParameterUpdated();
}

constructor(
    address _pulseRouterV1,
    address _pulseRouterV2,
    address _nineinchRouter,
    address _gelatoToken,
    uint256 _lockTime
) {
   ..........................
    require(_lockTime - block.timestamp <= 31536000, "Only 1 year can be initially set.");
    lockTime = _lockTime;
}
```

FOUND THREATS

Informational

Constructor missing Ownable(msg.sender) base constructor to set proper contract ownership to the deployer.

```
constructor(
    address _pulseRouterV1,
    address _pulseRouterV2,
    address _nineinchRouter,
    address _gelatoToken,
    uint256 _lockTime
) {
    pulseRouterV1 = IDEXRouter(_pulseRouterV1);
    pulseRouterV2 = IDEXRouter(_pulseRouterV2);
    nineinchRouter = IDEXRouter(_nineinchRouter);
    gelatoToken = IERC20(_gelatoToken);

    require(_lockTime - block.timestamp <= 31536000, "Only 1 year can be initially set.");
    lockTime = _lockTime;
}
```

- Recommendation:
  - Add Ownable(msg.sender) to set proper ownership to contract's deployer.

FOUND THREATS

Informational

Owner withdraw GELATO/WPLS pair on lock expiry but on withdrawTokens() comparison is made for GELATO/WETH pair.

```
function endLock() public onlyOwner {
.............................
    IDEXPair lpToken3 = IDEXPair(IDEXFactory(nineinchRouter.factory()).getPair(nineinchRouter.WPLS(), address(gelatoToken)));
.............................
}

function withdrawTokens(address _token) public onlyOwner {
.............................
    IDEXPair lpToken3 = IDEXPair(IDEXFactory(nineinchRouter.factory()).getPair(nineinchRouter.WETH(), address(gelatoToken)));
    require(_token != address(lpToken1) || _token != address(lpToken2) || _token != address(lpToken3), "You can not withdraw LP token.");
.............................
}
```

- Recommendation:
  - Correct with the right pair if this is not intentional.

FOUND THREATS

Informational (GELATO TOKEN)

If msg.value is less than contract's current balances, transaction will revert due to overflow.

```
function addLiquidity(uint256 gelAmount, uint256 deadline) public payable nonReentrant {
    _transferFrom(msg.sender, address(this), gelAmount);

    // Track starting balances
    uint256 plsSent = msg.value;
    uint256 gelBefore = balanceOf(address(this));

    // Add liquidity
    pulseRouterV2.addLiquidityETH{value: plsSent}(
        address(this),
        gelAmount,
        0,
        0,
        msg.sender,
        deadline
    );

    // Calculate the remaining balances
    uint256 plsUsed = plsSent - address(this).balance; // PLS used for liquidity
    uint256 plsRemaining = plsSent - plsUsed;
    uint256 gelRemaining = balanceOf(address(this)).sub(gelBefore);

    // Refund excess PLS if any
    if (plsRemaining > 0) {
        (bool success, ) = payable(msg.sender).call{
            value: plsRemaining,
            gas: 30000
        }("");
        require(success, "PLS transfer failed");
    }

    // Refund excess GEL tokens if any
    if (gelRemaining > 0) {
        _transferFrom(address(this), msg.sender, gelRemaining);
    }
}
```

Image 2

```
    uint256 plsSent = msg.value;
    uint256 plsBalancesBefore = address(this).balance;
    uint256 gelBefore = balanceOf(address(this));

    // Add liquidity
    pulseRouterV2.addLiquidityETH{value: plsSent}(
        address(this),
        gelAmount,
        0,
        0,
        msg.sender,
        deadline
    );

    uint256 plsBalanceAfter = address(this).balance;

    if(plsBalanceAfter > plsBalancesBefore) {
        uint256 remainingBalanceToSend = plsBalanceAfter - plsBalancesBefore;
        (bool success, ) = payable(msg.sender).call{
            value: remainingBalanceToSend,
            gas: 30000
        }("");
        require(success, "PLS transfer failed");
    }
```

- Recommendation:
  - Check contract's before and after balances and act accordingly if they are not the same.
  - Pseudo code sample provided for check of contract's native token balances (ETH, BNB, PLS etc.) in Image 2.

SPYWOLF

CRYPTO SECURITY

ABOUT US

We are a growing crypto security agency offering audits, KYCs and consulting services for some of the top names in the crypto industry.

- **OVER 700 SUCCESSFUL CLIENTS**
- **MORE THAN 1000 SCAMS EXPOSED**
- **MILLIONS SAVED IN POTENTIAL FRAUD**
- **PARTNERSHIPS WITH TOP LAUNCHPADS, INFLUENCERS AND CRYPTO PROJECTS**
- **CONSTANTLY BUILDING TOOLS TO HELP INVESTORS DO BETTER RESEARCH**

Audits | KYCs | dApps
Contract Development

To hire us, reach out to
contact@spywolf.co or
t.me/joe_SpyWolf

FIND US ONLINE

SPYWOLF.CO

@SPYWOLFNETWORK

@SPYWOLFNETWORK

Disclaimer

This report shows findings based on our limited project analysis, following good industry practice from the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, overall social media and website presence and team transparency details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report.
While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER:
By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice.
No one shall have any right to rely on the report or its contents, and SpyWolf and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (SpyWolf) owe no duty of care towards you or any other person, nor does SpyWolf make any warranty or representation to any person on the accuracy or completeness of the report.
The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and SpyWolf hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, SpyWolf hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SpyWolf, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts, website, social media and team.
No applications were reviewed for security. No product code has been reviewed.