SPYWOLF

Security Audit Report

Audit prepared for

NeverLetGo.AI

Completed on

May 29, 2024

KEY RESULTS

Cannot mint new tokens	N/A
Cannot pause trading (honeypot)	Passed
Cannot blacklist an address	Passed
Cannot raise taxes over 25%?	Passed
No proxy contract detected	Passed
Not required to enable trading	Passed
No hidden ownership	Passed
Cannot change the router	Passed
No cooldown feature found	Passed
Bot protection delay is lower than 5 blocks	Passed
Cannot set max tx amount below 0.05% of total supply	N/A
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

*Only new deposits/reinvestments can be paused

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

NEVER LET GO

PROJECT DESCRIPTION

According to their website:

Neverletgo.ai is a blockchain-powered adult entertainment platform designed to offer a unique user experience. Users can chat and interact with dream companions. Users can also opt to BECOME the characters complete with personal pages.

Release Date: Launched on April 24th, 2024

Category: Adult/AI

MainEngine

Token Name
N/A

Symbol
N/A

Contract Address
N/A

Network
N/A

Language
Solidity

Deployment Date
N/A

Contract Type
Game engine

Total Supply
N/A

Status
Not deployed

TAXES

Buy Tax
none

Sell Tax
none

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

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

No high risk-level threats found in this contract.

⚠ Medium Risk

No medium risk-level threats found in this contract.

⚠ Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

Informational

Owner can set luckywheel address.
Luckywheel address can grant free models unlock and free model discoveries to user.

```
modifier onlyLuckyWheel() {
    require(msg.sender == luckyWheelAddress, "Caller is not the Lucky Wheel");
    _;
}

function setLuckyWheelAddress(address _luckyWheelAddress) external onlyOwner {
    luckyWheelAddress = _luckyWheelAddress;
}

function grantBOGOUnlockOffer(address user) external onlyLuckyWheel {
    BOGOUnlockOffer storage offer = bogoUnlockOffers[user];
    offer.validUntil = block.timestamp + 3 days;
    offer.isUsed = false;
}

function grantFreeDiscoveries(address user, uint256 quantity) external onlyLuckyWheel {
    freeModelDiscoveries[user] += quantity;
}
```

FOUND THREATS

Informational

Owner can set whether model can receive usdc or not for models gift.

```
function setUsdcAcceptance(uint256 _modelId, bool _acceptUSDC) external onlyOwner {
    require(models[_modelId].exists, "Model ID does not exist");
    usdcAcceptanceByModelId[_modelId] = _acceptUSDC;
    emit UsdcAcceptanceSet(_modelId, _acceptUSDC);
}
```

Owner can set discovery costs for nft models.

```
function setDiscoverCost(uint256 _quantity, RarityCode _rarity, uint256 _usdcCost) external onlyOwner {
    require(_quantity > 0, "Quantity must be greater than zero");
    require(discoverCosts[_quantity][_rarity] == 0, "Discover cost already set for this quantity and rarity");
    discoverCosts[_quantity][_rarity] = _usdcCost;
}
```

FOUND THREATS

Informational

Owner can create new nft models with various rarities.

```
function createModel(uint256 _modelId, RarityCode _rarity, uint256 _cliffTimestamp) external onlyOwner {
    require(!models[_modelId].exists, "Model ID already exists");
    require(_cliffTimestamp >= block.timestamp, "Cliff timestamp must be in the future");

    models[_modelId].rarity = _rarity;
    models[_modelId].cliffTimestamp = _cliffTimestamp;
    models[_modelId].exists = true;

    emit ModelCreated(_modelId, _rarity, _cliffTimestamp);
}
```

Owner can set cost and earn amount threshold for each rarity.

```
function setRarity(RarityCode rarity, uint256 unlockCost, uint256 earnedAmountThreshold) external onlyOwner {
    rarities[rarity] = RarityInfo({
        unlockCost: unlockCost,
        earnedAmountThreshold: earnedAmountThreshold
    });
    emit RarityUpdated(rarity, unlockCost, earnedAmountThreshold);
}
```

Owner can set USDC to UWU conversion rate.
usdcToUWUEarnedRate is used in gift model functionality.

```
function setUSDCtoUWUEarnedRate(uint256 _usdcToUWURate) external onlyOwner {
    require(_usdcToUWURate > 0, "Conversion rate must be greater than zero");
    usdcToUWUEarnedRate = _usdcToUWURate;
    emit USDCtoUWUEarnedRateUpdated(_usdcToUWURate);
}
```

Dreamriders NFT

Token Name	Symbol
DreamridersNFT	N/A

Contract Address
0x851d231F11bAB6ae4D988A70A796B64A3c3cD6CF

Network	Language
Binance Smart Chain	Solidity

Deployment Date	Contract Type
May 17, 2024	NFT

Total Supply	Status
0	Launched

TAXES

Buy Tax
none

Sell Tax
none

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

FOUND THREATS

⚠ High Risk

No high risk-level threats found in this contract.

⚠ Medium Risk

No medium risk-level threats found in this contract.

⚠ Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

Informational

Owner can set base URI for NFTs,
If no URI is specified by user, the default base URI will be used.

```
function setBaseURI(string memory baseURI) public onlyOwner {
    require(!uriFrozen, "URI update is frozen.");
    baseTokenURI = baseURI;
}
```

Owner can change each minted NFT's URI until freezeBaseURI() is used.

```
function setTokenURI(uint256 tokenId, string memory uri) public onlyOwner {
    require(!uriFrozen, "Token URI updates are frozen.");
    require(_tokenExists(tokenId), "ERC721Metadata: URI set of nonexistent token");
    _tokenURIs[tokenId] = uri;
}
```

Owner can trigger freezeBaseURI() function.
Once freezeBaseURI() is used and uriFrozen is true, no further URI changes can be made either for base URI and individual NFT's URI.

```
function freezeBaseURI() public onlyOwner {
    uriFrozen = true;
}
```

LuckyWheel

Token Name	Symbol
N/A	N/A

Contract Address
N/A

Network	Language
N/A	Solidity

Deployment Date	Contract Type
N/A	Lotto

Total Supply	Status
N/A	Not Deployed

TAXES

Buy Tax	Sell Tax
none	**none**

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

FOUND THREATS

⚠ High Risk

First spin initiated from user will cost 333 UWU, every consecutive spin will cost to users 0.1 UWU.

```
function spin() external  {
    require(!hasPendingPrize(msg.sender), "Can't spin until pending prize is claimed.");
    uint256 spinCost = firstSpinDiscount[msg.sender] ? 1e17 : spinCostUWU; // 0.1 UWU or defined UWU cost
    firstSpinDiscount[msg.sender] = true;
.................
}
```

Recommendation:
Given that users' should have only one cheap spin at the cost of 0.1 UWU (user's first spin only), use the isEligibleForFirstSpinDiscount() functionality to user's status instead of direct comparison.

FOUND THREATS

⚠ Medium Risk

Anyone can call emergencyWithdrawUWU() functionality.
When emergencyWithdrawUWU is called, the contract sends the available UWU funds to the treasury address.
This might lead to undesired contract balances 'drain' into the treasury address from malicious actors, leaving no rewards for users in the contract and also causing future accounting problems for combined amount of _accumulatedTreasuryUWU + _accumulatedOwnersUWU + _accumulatedAirdropUWU if meanwhile users claim rewards from the contract.

```
function emergencyWithdrawUWU() external {
    uint256 contractBalance = uwuToken.balanceOf(address(this));
    uint256 reservedAmount = _accumulatedTreasuryUWU + _accumulatedOwnersUWU + _accumulatedAirdropUWU;

    require(contractBalance > reservedAmount, "No excess funds to withdraw");

    uint256 withdrawableAmount = contractBalance - reservedAmount;

    uwuToken.transfer(treasuryAddress, withdrawableAmount);

    emit EmergencyWithdrawal(treasuryAddress, withdrawableAmount, address(uwuToken));
}
```

Recommendation:
Apply access control to emergencyWithdrawUWU().
When emergency withdrawals are necessary, use withdrwaFunds() functionality first, then use the emergencyWithdrawUWU() to prevent accounting errors in _accumulatedTreasuryUWU + _accumulatedOwnersUWU + _accumulatedAirdropUWU.

FOUND THREATS

Informational

Owner can add new regular prize and mystery prize.

```
function addPrize(
    string memory description,
    PrizeAction action,
    uint8 rarity,
    uint256 quantity,
    uint256 probability,
    PrizeLimitType limitType
) external onlyOwner {
    prizes[prizeCount] =
    Prize(prizeCount, description, action, rarity, quantity, probability, limitType);
    totalProbability += probability;
    prizeCount++;
    emit PrizeAdded(prizeCount - 1, description);
}

function addMysteryPrize(
    string memory description,
    PrizeAction action,
    uint8 rarity,
    uint256 quantity,
    uint256 probability
) external onlyOwner {
    mysteryPrizes[mysteryPrizeCount] =
    MysteryPrize(mysteryPrizeCount, description, action, rarity, quantity, probability);
    mysteryTotalProbability += probability;
    mysteryPrizeCount++;
    emit MysteryPrizeAdded(mysteryPrizeCount - 1, description);
}
```

FOUND THREATS

Informational

Owner can remove regular prizes and mystery prizes.

```
function removePrize(uint256 prizeId) external onlyOwner {
    require(prizes[prizeId].originalIndex == prizeId, "Invalid prize ID");
    totalProbability -= prizes[prizeId].probability;
    delete prizes[prizeId];
    emit PrizeRemoved(prizeId);
}

function removeMysteryPrize(uint256 prizeId) external onlyOwner {
    require(mysteryPrizes[prizeId].originalIndex == prizeId, "Invalid prize ID");
    mysteryTotalProbability -= mysteryPrizes[prizeId].probability;
    delete mysteryPrizes[prizeId];
    emit MysteryPrizeRemoved(prizeId);
}
```

Owner can update cost for each wheel spin.
Current cost is 333 UWU for each spin.

```
uint256 public spinCostUWU = 333 * 1e18;
function updateSpinCost(uint256 newSpinCost) external onlyOwner {
    spinCostUWU = newSpinCost;
    emit SpinCostUpdated(newSpinCost);
}

function spin() external  {
    require(!hasPendingPrize(msg.sender), "Can't spin until pending prize is claimed.");
    uint256 spinCost = firstSpinDiscount[msg.sender] ? 1e17 : spinCostUWU;
.................
}
```

FOUND THREATS

Informational

Owner can withdraw any tokens from the contract except for BNB. When this function is present, in cases tokens are sent into the contract by mistake or purposefully, contract's owner can retrieve them.

```
function emergencyWithdrawToken(address tokenAddress) external {
    require(tokenAddress != address(uwuToken), "Use emergencyWithdrawUWU for UWU tokens");

    IERC20 token = IERC20(tokenAddress);
    uint256 tokenBalance = token.balanceOf(address(this));

    require(tokenBalance > 0, "No tokens to withdraw");

    token.transfer(treasuryAddress, tokenBalance);

    emit EmergencyWithdrawal(treasuryAddress, tokenBalance, tokenAddress);
}
```

WEBSITE

Website URL
https://neverletgo.ai/

Domain Registry
https://www.namecheap.com

Domain Expiration
N/A

Technical SEO Test
Passed

Security Test
Passed. SSL certificate present

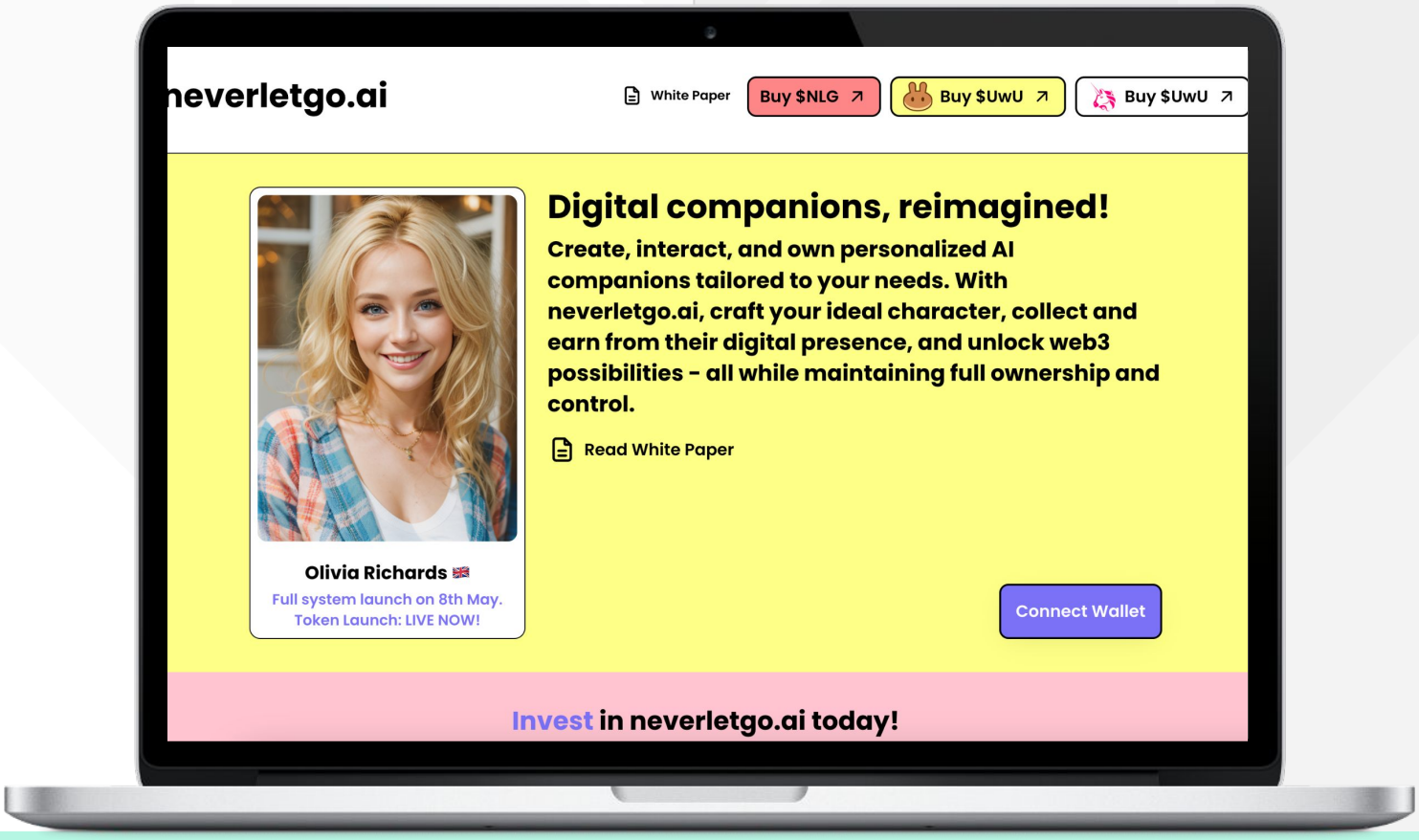
Design
Nice overall design with appropriate color scheme and graphics.

Content
Informative content. Users can understand what the project is about right away.

Whitepaper
Well written, explanatory.

Roadmap
No

Mobile-friendly?
Yes



neverletgo.ai

SOCIAL MEDIA

& ONLINE PRESENCE

ANALYSIS

Social media presence is new but active.

Twitter's X

@NLGFoundation

- 6,780 followers
- Responds to comments
- Daily posts

Discord

invite/TfEukaPhmN

- 4,471 members
- Active community

Telegram

@NLGFoundationChannel

- 3,009 subscribers
- Posts frequently

Medium

- Not available

SPYWOLF

CRYPTO SECURITY

Audits | KYCs | dApps
Contract Development

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**

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.