

SKELETON ECOSYSTEM

SMART CONTRACT AUDIT



1%ers
1%ers
ERC 20

0x6D7796c53Be8b9c8bc386a3851bE101C4e160b1E



Table of Contents

Table of Contents	1
Disclaimer	2
Overview	3
Creation/Audit Date	3
Verified Socials	3
Contract Functions Analysis	4
Contract Safety and Weakness	8
Detected Vulnerability Description	12
Contract Flow Graph	13
Contract Interaction Graph	14
Inheritance Graph	15
Contract Descriptions	16
Audit Scope	23

Global Disclaimer

This document serves as a disclaimer for the crypto smart contract audit conducted by Skeleton Ecosystem. The purpose of the audit was to review the codebase of the smart contracts for potential vulnerabilities and issues. It is important to note the following:

Limited Scope: The audit is based on the code and information available up to the audit completion date. It does not cover external factors, system interactions, or changes made after the audit. The audit itself can not guarantee 100% safety and can not detect common scam methods like farming and developer sell-out.

No Guarantee of Security: While we have taken reasonable steps to identify vulnerabilities, it is impossible to guarantee the complete absence of security risks or issues. The audit report provides an assessment of the contract's security as of the audit date.

Continued Development: Smart contracts and blockchain technology are evolving fields. Updates, forks, or changes to the contract post-audit may introduce new risks that were not present during the audit.

Third-party Code: If the smart contract relies on third-party libraries or code, those components were not thoroughly audited unless explicitly stated. Security of these dependencies is the responsibility of their respective developers.

Non-Exhaustive Testing: The audit involved automated analysis, manual review, and testing under controlled conditions. It is possible that certain vulnerabilities or issues may not have been identified.

Risk Evaluation: The audit report includes a risk assessment for identified vulnerabilities. It is recommended that the development team carefully reviews and addresses these risks to mitigate potential exploits.

Not Financial Advice: This audit report is not intended as financial or investment advice. Decisions regarding the use, deployment, or investment in the smart contract should be made based on a comprehensive assessment of the associated risks.

By accessing and using this audit report, you acknowledge and agree to the limitations outlined above. Skeleton Ecosystem and its auditors shall not be held liable for any direct or indirect damages resulting from the use of the audit report or the smart contract itself.

Please consult with legal, technical, and financial professionals before making any decisions related to the smart contract.

Overview

Contract Name	OnePercentersETH
Ticker/Symbol	1%ers
Blockchain	Ethereum ERC20
Contract Address	0x6D7796c53Be8b9c8bc386a3851bE101C4e160b1E
Creator Address	0x01A527cC1Bd1165f71458d5B6ff4FD9db0204BE7
Current Owner Address	0x00
Contract Explorer	https://etherscan.io/address/0x6d7796c53be8b9c8bc386a3851be101c4e160b1e#code
Compiler Version	v0.8.18+commit.87f61d96
License	MIT
Optimisation	Yes with 200 Runs
Total Supply	12,288 1%ers
Decimals	0

Creation/Audit

Contract Deployed	30.11.2023
Audit Created	03.01.2024
Audit Update	V 1.0

Verified Socials

Website	https://cryptosite.pro/onepercenterseth
Telegram	https://t.me/OnePercentersETH
Twitter (X)	https://x.com/eXtremeWhale

Contract Function Analysis



Pass



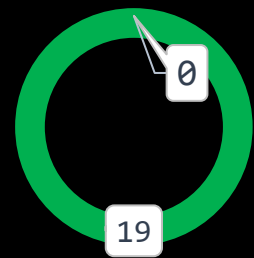
Attention Item











Risky Item

■ Pass

■ Attention



Contract Verified		The contract source code is uploaded to blockchain explorer and is open source, so everybody can read it.
Contract Ownership		0x00
Buy Tax	0 %	Shows the taxes for purchase transactions. Above 10% may be considered a high tax rate. More than 50% tax rate means may not be tradable. Fee can be set!
Sell Tax	5 %	Shows the taxes for sell transactions. Above 10% may be considered a high tax rate. More than 50% tax rate means may not be tradable. Fee can be set!
Honeypot Analyse		Holder is able to buy and sell. If honeypot: The contract blocks sell transfer from holder wallet. Multiple events may cause honeypot. Trading disabled, extremely high tax
Liquidity Status		Liquidity Locker status 02.01.2024: 100% LP Tokens Burned
Trading Disable Functions	 	Trading suspendable function found, but contract is renounced, this function can not be triggered. If a suspendable code is included, the token maybe neither be bought or sold (honeypot risk). If contract is renounced this function can't be used
Set Fees function	 	Fee Setting function found, but contract is renounced, this function can not be triggered. The contract owner may contain the authority to modify the transaction tax. If the transaction tax is increased to more than 49%, the tokens may not be able to be traded (honeypot risk).
Proxy Contract		Not a proxy contract!
Mint Function		No Mint Function detected Mint function is transparent or non-existent. Hidden mint functions may increase the amount of tokens in circulation and effect the price of the token. Owner can mint new tokens and sell.

Balance Modifier Function		<p>No Balance Modifier function found.</p> <p>If there is a function for this, the contract owner can have the authority to modify the balance of tokens at other addresses. For example revoke the bought tokens from the holders wallet. Common form of scam: You buy the token, but it's disappearing from your wallet.</p>
Blacklist Function		<p>No Blacklist Setting function found.</p> <p>If there is a blacklist, some addresses may not be able to trade normally. Example: you buy the token and right after your Wallet getting blacklisted. Like so you will be unable to sell. Honeypot Risk.</p>
Whitelist Function	 	<p>Whitelist Setting function found, but contract is renounced, this function can not be triggered.</p> <p>If there is a function for this Developer can set zero fee or no max wallet size for addresses (for example team wallets can trade without fee. Can cause farming)</p>
Hidden Owner Analysis		<p>No hidden or multi owner</p> <p>For contract with a hidden owner, developer can still manipulate the contract even if the ownership has been abandoned.</p>
Retrieve Ownership Function		<p>No functions found which can retrieve ownership of the contract.</p> <p>If this function exists, it is possible for the project owner to regain ownership even after relinquishing it. Also known as fake renounce.</p>
Self Destruct Function		<p>No Self Destruct function found.</p> <p>If this function exists and is triggered, the contract will be destroyed, all functions will be unavailable, and all related assets will be erased.</p>
Specific Tax Changing Function		<p>No Specific Tax Changing Functions found.</p> <p>If it exists, the contract owner may set a very outrageous tax rate for assigned address to block it from trading. Can assign all wallets at once!</p>
Trading Cooldown Function		<p>No Trading Cooldown Function found. If there is a trading cooldown function, the user will not be able to sell the token within a certain time or block after buying. Like a temporary honeypot.</p>
Max Transaction and Holding Modify Function	 	<p>Max Transaction and Holding Modify function found, but contract is renounced, this function can not be triggered.</p> <p>If there is a function for this, the maximum trading amount or maximum position can be modified. Can cause honeypot</p>
Transaction Limiting Function		<p>No Transaction Limiter Function Found.</p> <p>The number of overall token transactions may be limited (honeypot risk)</p>

Details of Risk - Attention Items



Function risks removed, based on the ownership of the contract got renounced and no option found to regain ownership.

Transaction Receipt Event Logs

149

Address 0x6d7796c53be8b9c8bc386a3851be101c4e160b1e

Name OwnershipTransferred (index_topic_1 address previousOwner, index_topic_2 address newOwner) [View Source](#)

Topics

0

0x8be0079c531659141344cd1fd0a4f28419497f9722a3daafe3b4186f6b6457e0

1: previousOwner

Dec ▾

→ 0x01A527cC1Bd1165f71458d5B6ff4FD9db0204BE7

2: newOwner

Dec ▾

→ 0x00

Data 0x

The following functions are listed for informational purposes.



Set Fee

Contract is renounced, this function can not be triggered.

The contract owner may contain the authority to modify the transaction tax. If the transaction tax is increased to more than 49%, the tokens may not be able to be traded

```

ftrace | funcSig
643 function updateBuyFees(
644     uint256 _treasuryFee,
645     uint256 _burnFee
646 ) external onlyOwner {
647     buyTreasuryFee = _treasuryFee;
648     buyBurnFee = _burnFee;
649     buyTotalFees = buyTreasuryFee + buyBurnFee;
650     require(buyTotalFees <= 20);
651 }
652
ftrace | funcSig
653 function updateSellFees(
654     uint256 _treasuryFee,
655     uint256 _burnFee
656 ) external onlyOwner {
657     sellTreasuryFee = _treasuryFee;
658     sellBurnFee = _burnFee;
659     sellTotalFees = sellTreasuryFee + sellBurnFee;
660     require(sellTotalFees <= 50);
661 }
  
```

⚠ Whitelist (Set excluded) Contract is renounced, this function can not be triggered.

If there is a function for this Developer can set zero fee or no max wallet size for addresses (for example team wallets can trade without fee)

```
ftrace | funcSig
663     function excludeFromFees(address account!, bool excluded!) public onlyOwner {
664         isExcludedFromFees[account!] = excluded!;
665         emit ExcludeFromFees(account!, excluded!);
666     }
667
```

⚠ Max Transaction and Holding Modify Function Contract is renounced, this function can not be triggered.

Max Transaction and Holding Modify function found.

If there is a function for this, the maximum trading amount or maximum position can be modified. Can cause honeypot

```
ftrace | funcSig
635     function updateMaxSwapTokens(
636         uint256 newAmount!
637     ) external onlyOwner returns (bool) {
638         require(newAmount! >= 1 && newAmount! <= 128);
639         maxSwapTokens = newAmount!;
640         return true;
641     }
642
```

⚠ Trading Suspensible Functions. Contract is renounced, this function can not be triggered.

```
604
605     function enableTrading() external onlyOwner {
606         tradingActive = true;
607         launchedAt = block.number;
608     }
609
```


Contract Security

Total Findings: 3

■ High 0

■ Medium 0

■ Low 2

■ Info 1



■ **High Severity Issues:** High possibility to cause problems, need to be resolved.

■ **Medium Severity Issue:** Will likely cause problems, recommended to resolve.

■ **Low Severity Issues:** Won't cause problems, but for improvement purposes could be adjusted.

■ **Informational Severity Issues:** Not harmful in any way, information for the developer team.

Contract Security

List of Found Issues

■ **High severity Issues: (0)**

■ **Medium severity issues: (0)**

■ **Low severity issues: (2)**

- Missing Events
- Front Running attack approval

■ **Informational severity issues: (1)**

- Public Functions Should be Declared External


Contract Weakness Classification


THE SMART CONTRACT WEAKNESS CLASSIFICATION REGISTRY (SWC REGISTRY) IS AN IMPLEMENTATION OF THE WEAKNESS CLASSIFICATION SCHEME PROPOSED IN EIP-1470. IT IS LOOSELY ALIGNED TO THE TERMINOLOGIES AND STRUCTURE USED IN THE COMMON WEAKNESS ENUMERATION (CWE) WHILE OVERLAYING A WIDE RANGE OF WEAKNESS VARIANTS THAT ARE SPECIFIC TO SMART CONTRACTS.

ID	Description	AI	Manual	Result
SWC-100	Function Default Visibility	Passed	Passed	Passed
SWC-101	Integer Overflow and Underflow	Passed	Passed	Passed
SWC-102	Outdated Compiler Version	Passed	Passed	Passed
SWC-103	Floating Pragma	Passed	Passed	Passed
SWC-104	Unchecked Call Return Value	Passed	Passed	Passed
SWC-105	Unprotected Ether Withdrawal	Passed	Passed	Passed
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed	Passed	Passed
SWC-107	Reentrancy	Passed	Passed	Passed
SWC-108	State Variable Default Visibility	Passed	Passed	Passed
SWC-109	Uninitialized Storage Pointer	Passed	Passed	Passed
SWC-110	Assert Violation	Passed	Passed	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed	Passed	Passed
SWC-112	Delegatecall to Untrusted Callee	Passed	Passed	Passed
SWC-113	DoS with Failed Call	Passed	Passed	Passed
SWC-114	Transaction Order Dependence	Passed	Passed	Passed
SWC-115	Authorization through tx.origin	Passed	Passed	Passed
SWC-116	Block values as a proxy for time	Passed	Passed	Passed
SWC-117	Signature Malleability	Passed	Passed	Passed
SWC-118	Incorrect Constructor Name	Passed	Passed	Passed
SWC-119	Shadowing State Variables	Passed	Passed	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed	Passed	Passed

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

Detected High and Medium Severity Vulnerability Description.

 No high or medium severity issues found

 Approve of front running attack (2 Items)

Item: 1	Location: Line 355-361	Severity:  Low
---------	------------------------	---

Function	<p>The <code>approve()</code> method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account. This can be abused by a token receiver when they try to withdraw certain tokens from the sender's account. Meanwhile, if the sender decides to change the amount and sends another approve transaction, the receiver can notice this transaction before it's mined and can extract tokens from both the transactions, therefore, ending up with tokens from both the transactions. This is a front-running attack affecting the ERC20 Approve function.</p> <p>The function <code>approve</code> can be front-run by abusing the <code>_approve</code> function.</p>
Remediation	<ol style="list-style-type: none"> 1. Introduce mechanisms that limit the maximum acceptable gas price for transactions. This can help prevent front-runners from drastically increasing the gas fees to prioritize their transactions. 2. Use transaction taxes to prevent against front-run attack

```

355  ftrace | funcSig
356  function approve(
357      address spender!,
358      uint256 amount!
359  ) public virtual override returns (bool) {
360      _approve(msgSender(), spender!, amount!);
361      return true;
362  }
  
```

Item: 2	Location: Line 363-380	Severity: ■ Low
---------	------------------------	--

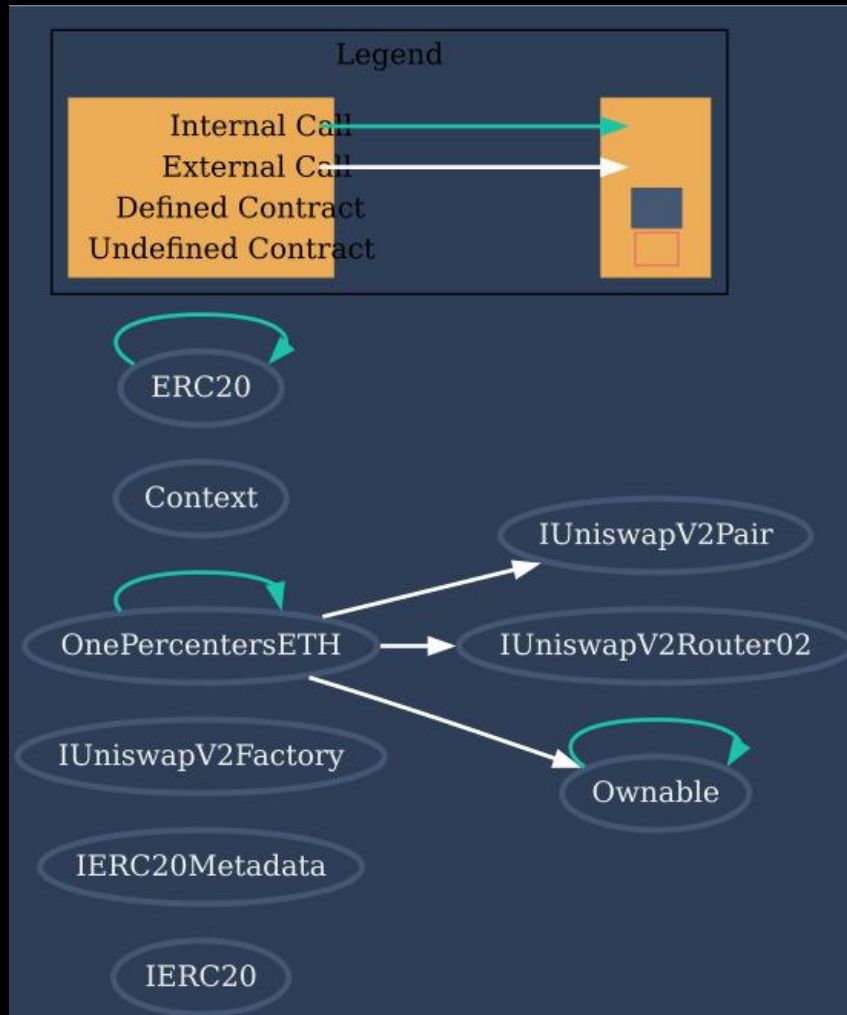
Function	<p>The <code>transferFrom()</code> method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account. This can be abused by a token receiver when they try to withdraw certain tokens from the sender's account. Meanwhile, if the sender decides to change the amount and sends another approve transaction, the receiver can notice this transaction before it's mined and can extract tokens from both the transactions, therefore, ending up with tokens from both the transactions. This is a front-running attack affecting the ERC20 Approve function.</p> <p>The function <code>transferFrom</code> can be front-run by abusing the <code>_approve</code> function.</p>
Remediation	<ol style="list-style-type: none"> 3. Introduce mechanisms that limit the maximum acceptable gas price for transactions. This can help prevent front-runners from drastically increasing the gas fees to prioritize their transactions. 4. Use transaction taxes to prevent against front-run attack

```

ftrace | funcSig
363     function transferFrom(
364         address sender!,
365         address recipient!,
366         uint256 amount!
367     ) public virtual override returns (bool) {
368         _transfer(sender!, recipient!, amount!);
369
370         uint256 currentAllowance = allowances[sender!][_msgSender()];
371         require(
372             currentAllowance >= amount!,
373             "ERC20: transfer amount exceeds allowance"
374         );
375         unchecked {
376             _approve(sender!, _msgSender(), currentAllowance - amount!);
377         }
378
379         return true;
380     }
381
  
```

[illegible]



































Contract Interaction Graph




























































































Inheritance Graph














































































Contract Functions






Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
IERC20	Interface			
L	totalSupply	External 		NO 
L	balanceOf	External 		NO 
L	transfer	External 		NO 
L	allowance	External 		NO 
L	approve	External 		NO 
L	transferFrom	External 		NO 
IERC20Metadata	Interface	IERC20		
L	name	External 		NO 
L	symbol	External 		NO 
L	decimals	External 		NO 
IUniswapV2Factory	Interface			
L	feeTo	External 		NO 
L	feeToSetter	External 		NO 
L	getPair	External 		NO 
L	allPairs	External 		NO 
L	allPairsLength	External 		NO 
L	createPair	External 		NO 

L	setFeeTo	External 		NO 
L	setFeeToSetter	External 		NO 
IUniswapV2Pair				
	Interface			
L	name	External 		NO 
L	symbol	External 		NO 
L	decimals	External 		NO 
L	totalSupply	External 		NO 
L	balanceOf	External 		NO 
L	allowance	External 		NO 
L	approve	External 		NO 
L	transfer	External 		NO 
L	transferFrom	External 		NO 
L	DOMAIN_SEPARATOR	External 		NO 
L	PERMIT_TYPEHASH	External 		NO 
L	nonces	External 		NO 
L	permit	External 		NO 
L	MINIMUM_LIQUIDITY	External 		NO 
L	factory	External 		NO 
L	token0	External 		NO 
L	token1	External 		NO 
L	getReserves	External 		NO 

L	price0CumulativeLast	External 		NO 
L	price1CumulativeLast	External 		NO 
L	kLast	External 		NO 
L	mint	External 		NO 
L	burn	External 		NO 
L	swap	External 		NO 
L	skim	External 		NO 
L	sync	External 		NO 
L	initialize	External 		NO 
IUniswapV2Router02	Interface			
L	factory	External 		NO 
L	WETH	External 		NO 
L	addLiquidity	External 		NO 
L	addLiquidityETH	External 		NO 
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External 		NO 
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External 		NO 
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External 		NO 

Context	Implementation			
L	_msgSender	Internal 		
L	_msgData	Internal 		
Ownable	Implementation	Context		
L		Public 		NO 
L	owner	Public 		NO 
L	renounceOwnership	Public 		onlyOwner
L	transferOwnership	Public 		onlyOwner
L	_transferOwnership	Internal 		
ERC20	Implementation	Context, IERC20, IERC20Metadata		
L		Public 		NO 
L	name	Public 		NO 
L	symbol	Public 		NO 
L	decimals	Public 		NO 
L	totalSupply	Public 		NO 
L	balanceOf	Public 		NO 
L	transfer	Public 		NO 
L	allowance	Public 		NO 
L	approve	Public 		NO 
L	transferFrom	Public 		NO 

L	increaseAllowance	Public 		NO 
L	decreaseAllowance	Public 		NO 
L	_transfer	Internal 		
L	_mint	Internal 		
L	_burn	Internal 		
L	_approve	Internal 		
L	_beforeTokenTransfer	Internal 		
L	_afterTokenTransfer	Internal 		
OnePercenter's ETH	Implementation	ERC20, Ownable		
L		Public 		ERC20
L		External 		NO 
L	addLiquidity	External 		onlyOwner
L	_addLiquidity	Private 		
L	enableTrading	External 		onlyOwner
L	removeLimits	External 		onlyOwner
L	wlContract	Public 		onlyOwner
L	excludeFromMaxTransaction	Public 		onlyOwner
L	updateSwapTokensAtAmount	External 		onlyOwner
L	updateMaxSwapTokens	External 		onlyOwner

L	updateBuyFees	External !		onlyOwner
L	updateSellFees	External !		onlyOwner
L	excludeFromFees	Public !		onlyOwner
L	setAutomatedMarketMakerPair	Public !		onlyOwner
L	_setAutomatedMarketMakerPair	Private 🔒		
L	updateMarketingWallet	External !		onlyOwner
L	_transfer	Internal 🔒		
L	swapBack	Private 🔒		
L	swapTokensForEth	Private 🔒		
L	autoBurnLiquidityPairTokens	Internal 🔒		



Function
can modify
state



Function
is payable

Audit Scope

Audit Method.

Our smart contract audit is an extensive methodical examination and analysis of the smart contract's code that is used to interact with the blockchain. Goal: discover errors, issues and security vulnerabilities in the code. Findings getting reported and improvements getting suggested.

Automatic and Manual Review

We are using automated tools to scan functions and weaknesses of the contract. Transfers, integer over-undeflow checks such as all CWE events.

Tools we use:

Visual Studio Code

CWE

SWC

Solidity Scan

SVD

In manual code review our auditor looking at source code and performing line by line examination. This method helps to clarify developer's coding decisions and business logic.

Skeleton Ecosystem

<https://skeletonecosystem.com>

<https://github.com/SkeletonEcosystem/Audits>

