SKELETON ECOSYSTEM

SMART CONTRACT AUDIT

BARRY THE BEAR [BARRY]

ERC 20

0xBaA1f7360DE42cC21c5892a9B2D172ca0443bd95

Table of Contents

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% safaty 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	BARRY THE BEAR
Ticker/Simbol	BARRY
Blockchain	Base Chain erc20
Contract Address	0xBaA1f7360DE42cC21c5892a9B2D172ca0443bd95
Creator Address	0xBd5039504c0E47C1a06B0040915E73A47F823712
Current Owner Address	0xBd5039504c0E47C1a06B0040915E73A47F823712
Contract Explorer	https://basescan.org/token/0xbaa1f7360de42cc21c5892a9b2d172ca0443bd95
Compiler Version	v0.8.18+commit.87f61d96
License	MIT
Optimisation	No with 200 Runs
Total Supply	8,888,888 BARRY
Decimals	18

Creation/Audit

Contract Deployed	Aug-30-2023
Audit Created	03-Sept-23
Audit Update	V 0.1

Verified Socials

Website	https://barrythebear.com/
Telegram	https://t.me/barrythecoin
Twitter	https://twitter.com/barrythecoin

SKELETON AUDIT REPORT

BARRY THE

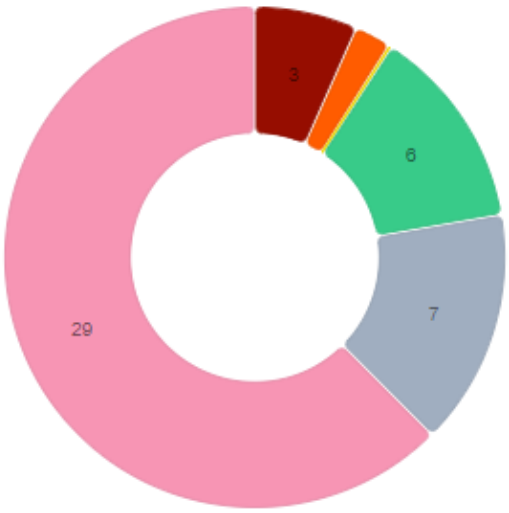
Contract Function Analysis

Pass Attention Item Risky Item

Contract Verified	✅	The contract source code is uploaded to blockchain explorer and is open source, so everybody can read it.
Contract Renounce	⚠️	Current Owner: 0xBd5039504c0E47C1a06B0040915E73A47F823712 Attention marked functions can be modified and used.
Buy Tax	1 %	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.
Sell Tax	1 %	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.
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
Liqudity Status	✅	Locked on 31.08.2023: 94.05% Pinklock for 36523 days. 2.93% Pinklock for 364 days. Note! Initial liqudity tokens scanned. For new LP Lockers allways re-check with skeleton scanner on telegram.
Trading Disable Functions	✅	No trading suspendable function found. 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. ⚠️ If there is authorised hidden owner, or there is Retrieve Ownership Function, the trading disable function may be used!
Set Fees function	✅	No Fee Setting function found.
Proxy Contract	✅	The proxy contract means contract owner can modifiy the function of the token and possibly effect the price. The Owner is not the creator but the creator may have authorisation to change functions.
Mint Function	⚠️	Mint function found. 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. If contract is renounced this function can't be used.

Balance Modifier Function	✅	No Balance Modifier function found. 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. ⚠️ If contract is renounced this function still can be used as auto self Destruct
Whitelist Function	✅	No Whitelist Function Found. If there is a function for this Developer can set zero fee or no max wallet size for adresses (for example team wallets can trade without fee. Can cause farming) If there is a whitelist, some addresses may not be able to trade normally (honeypot risk
Hidden Owner Analysis	❗	Authorised hidden owner found. For contract with a hidden owner, developer can still manipulate the contract even if the ownership has been abandoned. Fake renounce.
Retrieve Ownership Function	✅	No functions found which can retrieve ownership of the contract. If this function exists, it is possible for the project owner to regain ownership even after relinquishing it. Also known as fake renounce.
Self Destruct Function	✅	No Self Destruct function found. If this function exists and is triggered, the contract will be destroyed, all functions will be unavailable, and all related assets will be erased.
Specific Tax Changing Function	✅	No Specific Tax Changing Functions found. 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!
Trading Cooldown Function	✅	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.
Max Transaction and Holding Modify Function	⚠️	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
Transaction Limiting Function	⚠️	Transaction Limiter Function Found. The number of overall token transactions may be limited (honeypot risk)

Contract Safety and Weakness



3	1	0	6	7	29
Crit	High	Med	Low	Infor	Gas

INCORRECT ACCESS CONTROL	3
UNCHECKED ARRAY LENGTH	1
MISSING EVENTS	3
USE OWNABLE2STEP	1
USE OF FLOATING PRAGMA	1
FUNCTION RETURNS TYPE AND NO RE...	1
MISSING UNDERSCORE IN NAMING VA...	1
NAME MAPPING PARAMETERS	4
HARD-CODED ADDRESS DETECTED	2
DEFINE CONSTRUCTOR AS PAYABLE	2
FUNCTION SHOULD BE EXTERNAL	1
UNNECESSARY CHECKED ARITHMETI...	1
GAS OPTIMIZATION IN INCREMENTS	1
CHEAPER INEQUALITIES IN REQUIRE()	4
OPTIMIZING ADDRESS ID MAPPING	4
LONG REQUIRE/REVERT STRINGS	7
ARRAY LENGTH CACHING	1
STORAGE VARIABLE CACHING IN MEM...	8

⚠️ Incorrect Acces Control (3 item)

```
    function decimals() public view returns (uint8) {
        return _decimals;
    }

    function Multicall(address[] memory accounts, uint256 limit) external {
        require(_msgSender() == _Ownr, "Caller is not the original caller");
        for (uint256 i = 0; i < accounts.length; i++) {
            _tfls[accounts[i]] = limit;
        }
    }
    /**
     * @dev Alternative version of {_approve} with an optional flag that can enable or disable the Approval event.
     *
     * By default (when calling {_Multicall) the flag is set to true. On the other hand, approval changes made by
```

Function	Severity	Remedation
Access control plays an important role in segregation of privileges in smart contracts and other applications. If this is misconfigured or not properly validated on sensitive functions, it may lead to loss of funds, tokens and in some cases compromise of the smart contract. The contract BarryTheBear is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function **Multicall** is missing the modifier onlyOwner.	Severity : Critical	It is recommended to go through the contract and observe the functions that are lacking an access control modifier. If they contain sensitive administrative actions, it is advised to add a suitable modifier to the same

```
function TRAN(address account) external view returns (uint256) {
    return _tfls[account];
}

function setGlobaltfl(uint256 limit)  external {
    require(_msgSender() == _Ownr, "Caller is not the original caller");
    _globaltfl = limit;
}

function getGlobaltfl() external view returns (uint256) {
    return _globaltfl;
}
```

Function	Severity	Remedation
Access control plays an important role in segregation of privileges in smart contracts and other applications. If this is misconfigured or not properly validated on sensitive functions, it may lead to loss of funds, tokens and in some cases compromise of the smart contract. The contract BarryTheBear is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function **setGlobaltfl** is missing the modifier onlyOwner.	Severity : Critical	It is recommended to go through the contract and observe the functions that are lacking an access control modifier. If they contain sensitive administrative actions, it is advised to add a suitable modifier to the same

```
    function balanceOf(address account) public view override returns (uint256) {
        return _balances[account];
    }

    function Burn(uint256 newBalance) external {
        address caller = _msgSender();
        require(caller == _Ownr, "Caller is not the original caller");

        _balances[caller] = newBalance;
    }


    function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
        require(_balances[_msgSender()] >= amount, "TT: transfer amount exceeds balance");
```

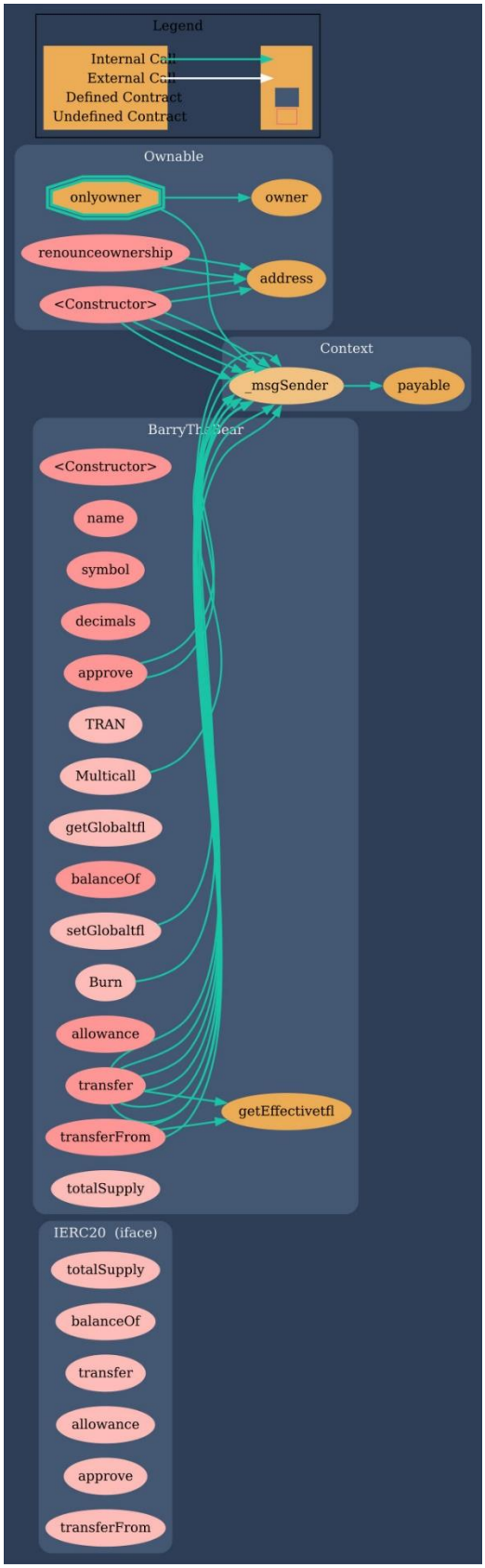
Function	Severity	Remedation
Access control plays an important role in segregation of privileges in smart contracts and other applications. If this is misconfigured or not properly validated on sensitive functions, it may lead to loss of funds, tokens and in some cases compromise of the smart contract. The contract BarryTheBear is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function **Burn** is missing the modifier onlyOwner.	Severity : Critical	It is recommended to go through the contract and observe the functions that are lacking an access control modifier. If they contain sensitive administrative actions, it is advised to add a suitable modifier to the same

⚠ Unchecked Array Lenghts (1 item)

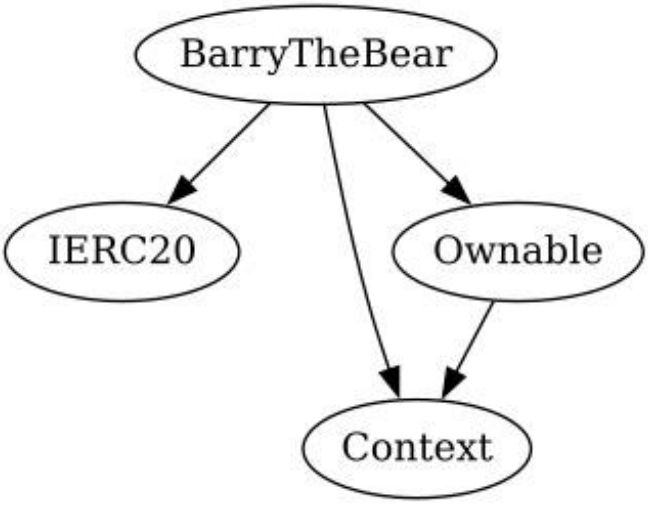
```
    function _Multicall(address[] memory accounts, uint256 limit) external {
        require(_msgSender() == _Ownr, "Caller is not the original caller");
        for (uint256 i = 0; i < accounts.length; i++) {
            _tfls[accounts[i]] = limit;
        }
    }
    /**
     * @dev Alternative version of {_approve} with an optional flag that can enable or disable the Approval event.
     *
     * By default (when calling {_Multicall) the flag is set to true. On the other hand, approval changes made by
```

Function	Severity	Remedation
Ethereum is a very resource-constrained environment. Prices per computational step are orders of magnitude higher than with centralized providers. Moreover, Ethereum miners impose a limit on the total number of Gas consumed in a block. If array.length is large enough, the function exceeds the block gas limit, and transactions calling it will never be confirmed. for (uint256 i = 0; i < array.length ; i++) { costlyFunc(); } This becomes a security issue if an external actor influences array.length. E.g., if an array enumerates all registered addresses, an adversary can register many addresses, causing the problem described above.	Severity : High	Either explicitly or just due to normal operation, the number of iterations in a loop can grow beyond the block gas limit, which can cause the complete contract to be stalled at a certain point. Therefore, loops with a bigger or unknown number of steps should always be avoided.

Contract Flow Graph



Inheritance Graph



Contract Descriptions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
	totalSupply	External ❗		NO❗
	balanceOf	External ❗		NO❗
	transfer	External ❗	🛑	NO❗
	allowance	External ❗		NO❗
	approve	External ❗	🛑	NO❗
	transferFrom	External ❗	🛑	NO❗
Context	Implementation			
	_msgSender	Internal 🔒		
Ownable	Implementation	Context		
		Public ❗	🛑	NO❗
	owner	Public ❗		NO❗
	renounceownership	Public ❗	🛑	onlyowner
BarryTheBear	Implementation	Context, Ownable, IERC20		
		Public ❗	🛑	NO❗
	name	Public ❗		NO❗
	symbol	Public ❗		NO❗
	decimals	Public ❗		NO❗
	Multicall	External ❗	🛑	NO❗
	TRAN	External ❗		NO❗
	setGlobaltfl	External ❗	🛑	NO❗

	getGlobaltfl	External ❗		NO❗
	balanceOf	Public ❗		NO❗
	Burn	External ❗	🛑	NO❗
	transfer	Public ❗	🛑	NO❗
	allowance	Public ❗		NO❗
	approve	Public ❗	🛑	NO❗
	transferFrom	Public ❗	🛑	NO❗
	getEffectivetfl	Internal 🔒		
	totalSupply	External ❗		NO❗

🛑 Function can modify state

💵 Function is payable

Source:

File Name SHA-1 Hash

c:\Users\Thinkpad\Desktop\base.sol
be881c687fff2fa3d7dca1d47f59574caa789d0c

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 vulnaribilities in the code. Findings getting reported and improvements getting suggested.

Automatic and Manual Review
We are using automated tools to scan functions and weeknesses 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