Cyberscope

Audit Report

ETHnothing

June 2023

SHA256 004e2a7d174240fdfa951a6cd2afd325bf4a135e5c27759435b217086c114179

Audited by © cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Unresolved
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	EFLC	Exclude Function Logic Concern	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	PAV	Pair Address Validation	Unresolved
●	RBFF	Redundant Buy Fee Functionality	Unresolved
●	RE	Redundant Event	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Review

Contract Name	Nothing
Testing Deploy	https://testnet.bscscan.com/address/0x5467acdc3d36dcb2587491b1dd24686cf636da67
Symbol	TST
Decimals	8
Total Supply	1.000.000.000

Audit Updates

Initial Audit	01 Jun 2023 https://github.com/cyberscope-io/audits/blob/main/ETHnothing/v1/audit.pdf
Corrected Phase 2	22 Jun 2023

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	9353af89436556f7ba8abb3f37a6677249a a4df6024fbfaa94f79ab2f44f3231
@openzeppelin/contracts/token/ERC20/ERC20.sol	bce14c3fd3b1a668529e375f6b70ffdf9cef 8c4e410ae99608be5964d98fa701
@openzeppelin/contracts/token/ERC20/extensions /ERC20Burnable.sol	0344809a1044e11ece2401b4f7288f414ea 41fa9d1dad24143c84b737c9fc02e
@openzeppelin/contracts/token/ERC20/extensions /IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166 689e55dc037a7f2f790d057811990
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db800 3d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a2 3a4baa0b5bd9add9fb6d6a1549814a
contracts/Nothing.sol	004e2a7d174240fdfa951a6cd2afd325bf4 a135e5c27759435b217086c114179

Findings Breakdown

Critical	1
Medium	0
Minor / Informative	7

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	1	0	0	0
Medium	0	0	0	0
Minor / Informative	7	0	0	0

MT - Mints Tokens

Criticality	Critical
Location	contracts/Nothing.sol#L97
Status	Unresolved

Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the `mint` function. As a result, the contract tokens will be highly inflated.

```
function mint(address account, uint256 amount) public onlyOwner {
  _mint(account, amount);
}
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

EFLC - Exclude Function Logic Concern

Criticality	Minor / Informative
Location	contracts/Nothing.sol#L
Status	Unresolved

Description

The contract implements an exclusion mechanism to exempt addresses from being charged fees. However, the implementation only checks if the sender `from` address is excluded from fees and not recipient `to` too. This inconsistency can lead to unintended fee deductions for recipients who should be exempted from fees.

```
bool isExcludedFromFees = excludedFromFees[from];
if (!isExcludedFromFees) {
    if (isSell) {
        taxAmount = (amount * sellFee / 10_000);
    } else if (isBuy) {
        taxAmount = (amount * buyFee / 10_000);
    }
}
```

Recommendation

It is recommended to check the exclusion status of both recipient and sender addresses before deducting any fees.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	contracts/Nothing.sol#L39,49,60
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract updates the variables even if its current state is the same as the one passed as an argument. As a result, the contract performs redundant storage writes.

```
function setFeeReceiver(address receiver) public onlyOwner {
    require(receiver != address(0), "Receiver can't be address zero");
    feeReceiver = receiver;
    emit FeeReceiverAddressUpdate(receiver);
}

function setRewardReceiver(address receiver) public onlyOwner {
    require(receiver != address(0), "Receiver can't be address zero");
    rewardReceiver = receiver;
    emit RewardFeeReceiverAddressUpdate(receiver);
}

function setMarketPair(address pair, bool state) public onlyOwner {
    require(pair != address(0), "Pair can't be address zero");
    marketPairs[pair] = state;
    emit MarketPairUpdated(pair, state);
}
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

PAV - Pair Address Validation

Criticality	Minor / Informative
Location	contracts/Nothing.sol#L60
Status	Unresolved

Description

The contract is missing address validation in the pair address argument. The absence of validation reveals a potential vulnerability, as it lacks proper checks to ensure the integrity and validity of the pair address provided as an argument. The pair address is a parameter used in certain methods of decentralized exchanges for functions like token swaps and liquidity provisions.

The absence of address validation in the pair address argument can introduce security risks and potential attacks. Without proper validation, if the owner's address is compromised, the contract may lead to unexpected behavior like loss of funds.

```
function setMarketPair(address pair, bool state) public onlyOwner {
    require(pair != address(0), "Pair can't be address zero");
    marketPairs[pair] = state;
    emit MarketPairUpdated(pair, state);
}
```

Recommendation

To mitigate the risks associated with the absence of address validation in the pair address argument, it is recommended to implement comprehensive address validation mechanisms. A recommended approach could be to verify pair existence in the decentralized application. Prior to interacting with the pair address contract, perform checks to verify the existence and validity of the contract at the provided address. This can be achieved by querying the provider's contract or utilizing external libraries that provide contract verification services.

RBFF - Redundant Buy Fee Functionality

Criticality	Minor / Informative
Location	contracts/Nothing.sol#L131
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The buy fee is fixed to zero and it can not be mutated. Hence the buy fee functionality is redundant.

```
uint16 public buyFee = 0; // Buy fee ratio (Default: 0%)

} else if (isBuy) {
    taxAmount = (amount * buyFee / 10_000);
}
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it. It is recommended to remove redundant functionality.

RE - Redundant Event

Criticality	Minor / Informative
Location	contracts/Nothing.sol#L26
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The event `FeeRatiosUpdated` is not utilized in the contract's implementation. Hence, it is redundant.

```
event FeeRatiosUpdated(uint16 buyFee, uint16 sellFee);
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it. It is recommended to remove redundant events.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	contracts/Nothing.sol#L16,17,20
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
uint16 public sellFee = 500
uint16 public buyFee = 0
uint8 private _decimals = 8
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/Nothing.sol#L3
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.17;
```

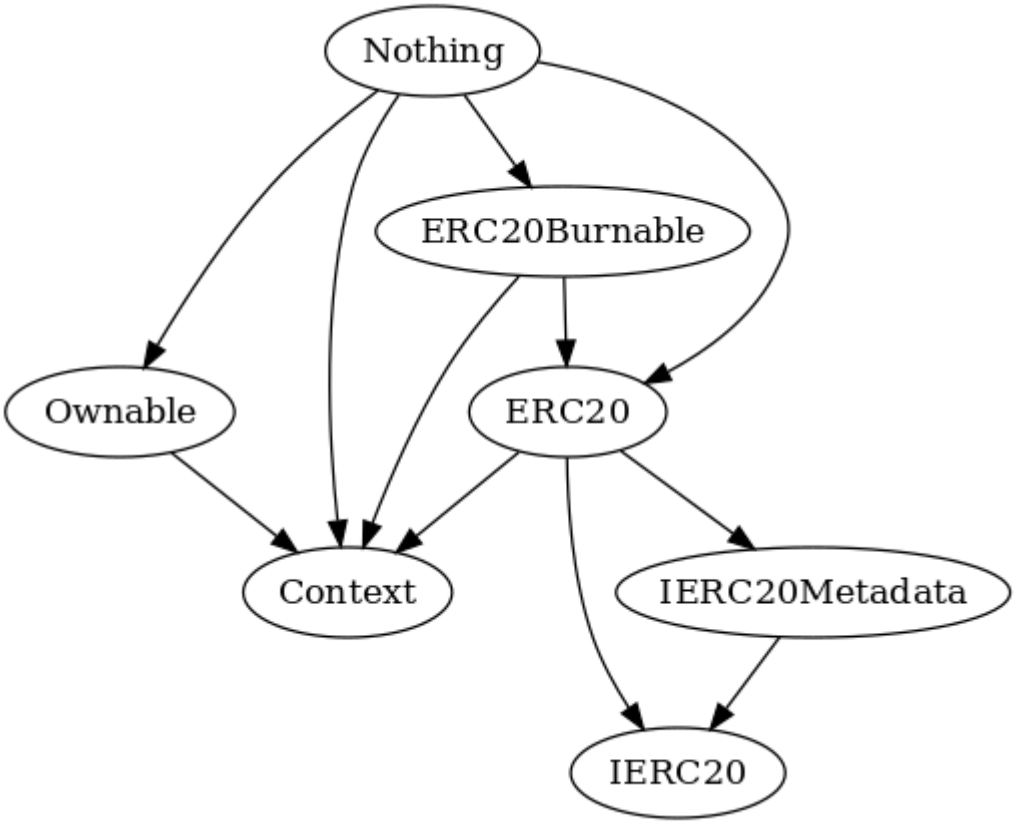
Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

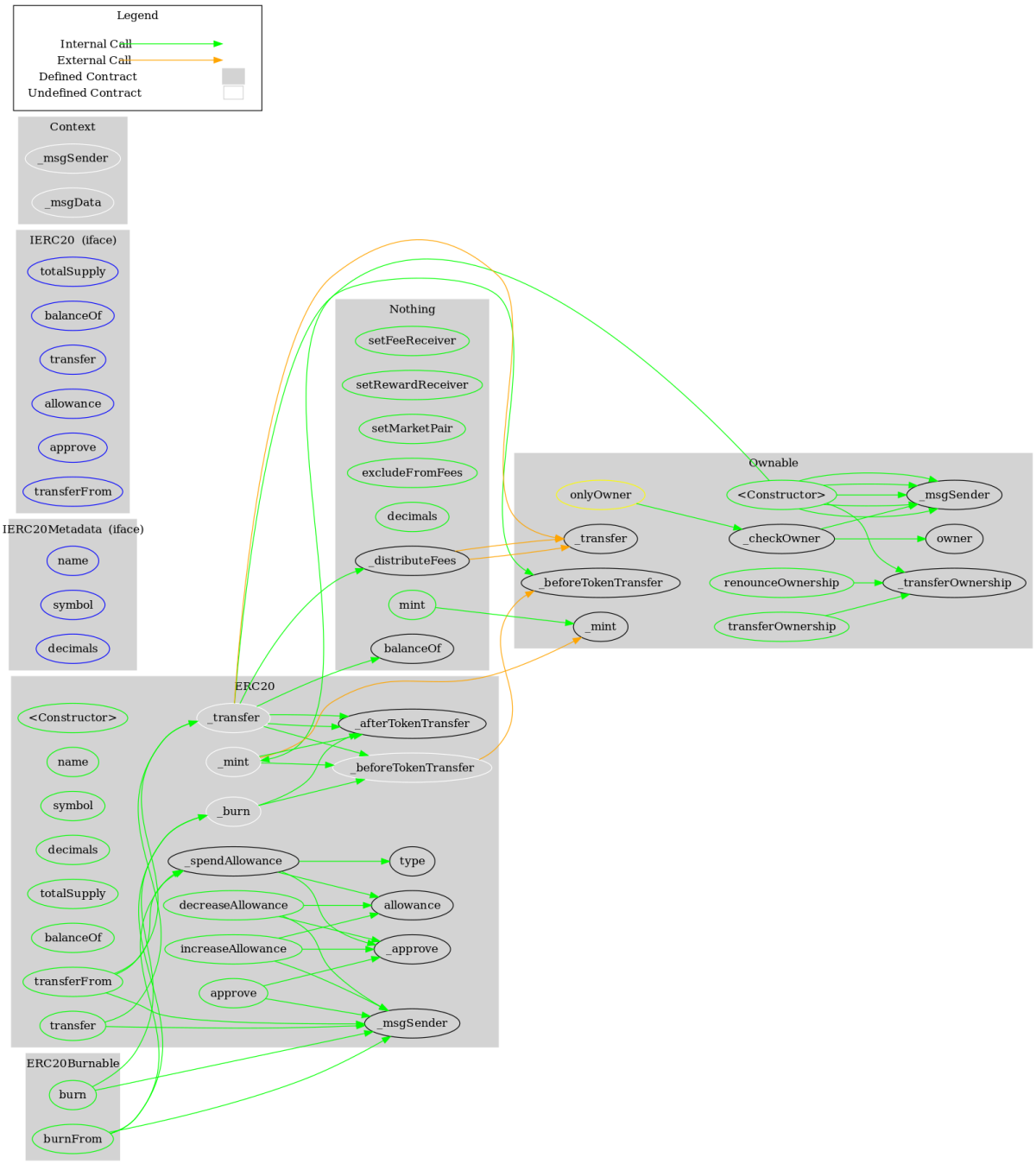
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Nothing	Implementation	Context, ERC20, ERC20Burnable, Ownable		
		Public	✓	ERC20
	setFeeReceiver	Public	✓	onlyOwner
	setRewardReceiver	Public	✓	onlyOwner
	setMarketPair	Public	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	decimals	Public		-
	_mint	Internal	✓	
	mint	Public	✓	onlyOwner
	_beforeTokenTransfer	Internal	✓	
	_transfer	Internal	✓	
	_distributeFees	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Context
_msgSender
_msgData
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
IERC20Metadata (iface)
name
symbol
decimals
Nothing
setFeeReceiver
setRewardReceiver
setMarketPair
excludeFromFees
decimals
_distributeFees
mint
balanceOf
Ownable
onlyOwner
<Constructor>
_msgSender
_transfer
_checkOwner
owner
_beforeTokenTransfer
renounceOwnership
_transferOwnership
_mint
transferOwnership
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transferFrom
transfer
_transfer
_mint
_burn
_afterTokenTransfer
_beforeTokenTransfer
_spendAllowance
decreaseAllowance
increaseAllowance
approve
type
allowance
_approve
_msgSender
ERC20Burnable
burn
burnFrom

Summary

ETHnothing contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. There are some functions that can be abused by the owner like mint tokens. if the contract owner abuses the mint functionality, then the contract will be highly inflated. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. There is also a limit of max 5% fee.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io