Cyberscope

Audit Report

Navy Seal

August 2023

Network	ETH
Address	0x34df29dd880e9fe2cec0f85f7658b75606fb2870
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	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	FSA	Fixed Swap Address	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved

Table of Contents

Review

Contract Name	NAVYSEALCON
Compiler Version	v0.8.20+commit.a1b79de6
Optimization	200 runs
Explorer	https://etherscan.io/address/0x34df29dd880e9fe2cec0f85f7658b75606fb2870
Address	0x34df29dd880e9fe2cec0f85f7658b75606fb2870
Network	ETH
Symbol	NAVYSEAL
Decimals	9
Total Supply	10,000,000,000

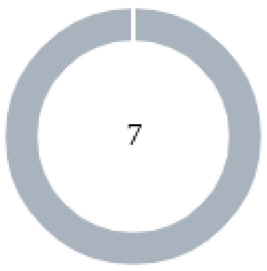
Audit Updates

Initial Audit	21 Aug 2023

Source Files

Filename	SHA256
NAVYSEALCON.sol	d9fee17873cd39101e26f08520a368dd87782ddfefa0ea31d158eb7e3691efd3

Findings Breakdown



●	Critical	0
●	Medium	0
●	Minor / Informative	7

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

FSA - Fixed Swap Address

Criticality	Minor / Informative
Location	NAVYSEALCON.sol#L315
Status	Unresolved

Description

The swap address is assigned once and it can not be changed. It is a common practice in decentralized exchanges to create new swap versions. A contract that cannot change the swap address may not be able to catch up to the upgrade. As a result, the contract will not be able to migrate to a new liquidity pool pair or decentralized exchange.

```
uniswapV2Router =
IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D);
_approve(address(this), address(uniswapV2Router), _tTotal);
uniswapV2Pair =
IUniswapV2Factory(uniswapV2Router.factory()).createPair(address(t
his), uniswapV2Router.WETH());
uniswapV2Router.addLiquidityETH{value:
address(this).balance}(address(this),balanceOf(address(this)),0,0
,owner(),block.timestamp);
IERC20(uniswapV2Pair).approve(address(uniswapV2Router),
type(uint).max);
```

Recommendation

The team is advised to add the ability to change the pair and router address in order to cover potential liquidity pool migrations. It would be better to support multiple pair addresses so the token will be able to have the same behavior in all the decentralized liquidity pairs.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	NAVYSEALCON.sol#L334
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The `manualSwap` function allows the `_taxWallet` to manually trigger the swap of the contract's token balance for Ether (ETH). Subsequently, any ETH balance held by the contract is sent to the `_taxWallet` address.

It is important to note that the price of the token representing it, can be highly volatile. If the contract holds a significant amount of tokens, invoking the `manualSwap` function will result in a large swap of tokens for ETH. This could lead to significant price volatility. Such a large swap could adversely impact the token's price, potentially leading to financial implications for the parties involved.

```
   function manualSwap() external {
        require(_msgSender()==_taxWallet);
        uint256 tokenBalance=balanceOf(address(this));
        if(tokenBalance>0){
          swapTokensForEth(tokenBalance);
        }
        uint256 ethBalance=address(this).balance;
        if(ethBalance>0){
          sendETHToFee(ethBalance);
        }
    }
```

Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the total supply. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	NAVYSEALCON.sol#L313,325
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
    function openTrading() external onlyOwner() {
        require(!tradingOpen,"trading is already open");
        uniswapV2Router =
IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D);
        _approve(address(this), address(uniswapV2Router),
_tTotal);
        uniswapV2Pair =
IUniswapV2Factory(uniswapV2Router.factory()).createPair(address(t
his), uniswapV2Router.WETH());
        uniswapV2Router.addLiquidityETH{value:
address(this).balance}(address(this),balanceOf(address(this)),0,0
,owner(),block.timestamp);
        IERC20(uniswapV2Pair).approve(address(uniswapV2Router),
type(uint).max);
        swapEnabled = true;
        tradingOpen = true;
    }

    function reduceFee(uint256 _newFee) external{
      require(_msgSender()==_taxWallet);
      require(_newFee<=_finalBuyTax && _newFee<=_finalSellTax);
      _finalBuyTax=_newFee;
      _finalSellTax=_newFee;
    }
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	NAVYSEALCON.sol
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	NAVYSEALCON.sol#L162
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
_taxWallet
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	NAVYSEALCON.sol#L130,131,134,135,136,145,146
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.

```
uint256 private _initialBuyTax=20
uint256 private _initialSellTax=20
uint256 private _reduceBuyTaxAt=10
uint256 private _reduceSellTaxAt=20
uint256 private _preventSwapBefore=20
uint256 public _taxSwapThreshold= 100000000 * 10**_decimals
uint256 public _maxTaxSwap= 100000000 * 10**_decimals
```

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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	NAVYSEALCON.sol#L109,139,140,141,142,143,144,145,146,325
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
function WETH() external pure returns (address);
uint8 private constant _decimals = 9
uint256 private constant _tTotal = 10000000000 * 10**_decimals
string private constant _name = unicode"What the fuck did you
just fucking say about me, you little bitch? I'll have you know
I graduated top of my class in the Navy Seals, and I've been
involved in numerous secret raids on Al-Quaeda, and I have over
300 confirmed kills. I am trained in gorilla warfare and I'm
the top sniper in the entire US armed forces. You are nothing
to me but just another target. I will wipe you the fuck out
with precision the likes of which has never been seen before on
this Earth, mark my fucking words. You think you can get away
with saying that shit to me over the Internet? Think again,
fucker. As we speak I am contacting my secret network of spies
across the USA and your IP is being traced right now so you
better prepare for the storm, maggot. The storm that wipes out
the pathetic little thing you call your life. You're fucking
dead, kid. I can be anywhere, anytime, and I can kill you in
over seven hundred ways, and that's just with my bare hands.
Not only am I extensively trained in unarmed combat, but I have
access to the entire arsenal of the United States Marine Corps
and I will use it to its full extent to wipe your miserable ass
off the face of the continent, you little shit. If only you
could have known what unholy retribution your little clever
comment was about to bring down upon you, maybe you would have
held your fucking tongue. But you couldn't, you didn't, and now
you're paying the price, you goddamn idiot. I will shit fury
all over you and you will drown in it. You're fucking dead,
kiddo."
string private constant _symbol = unicode"NAVYSEAL"
uint256 public _maxTxAmount = 200000000 * 10**_decimals
uint256 public _maxWalletSize = 200000000 * 10**_decimals
uint256 public _taxSwapThreshold= 100000000 * 10**_decimals
uint256 public _maxTaxSwap= 100000000 * 10**_decimals
uint256 _newFee
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.
Find more information on the Solidity documentation
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

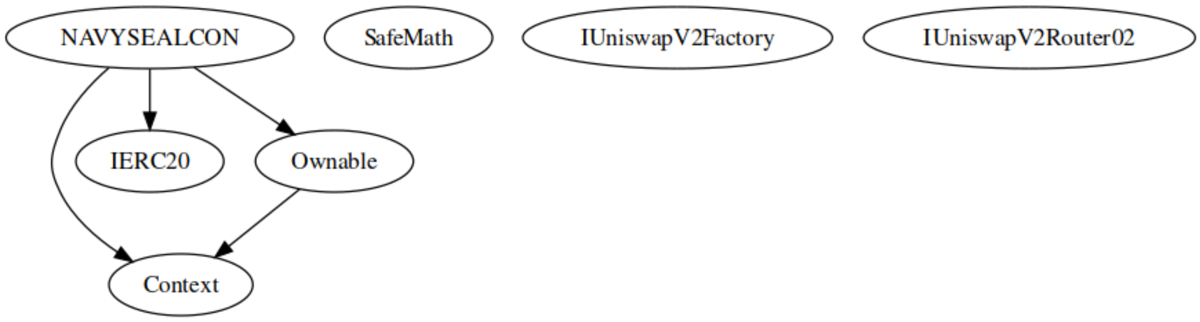
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		

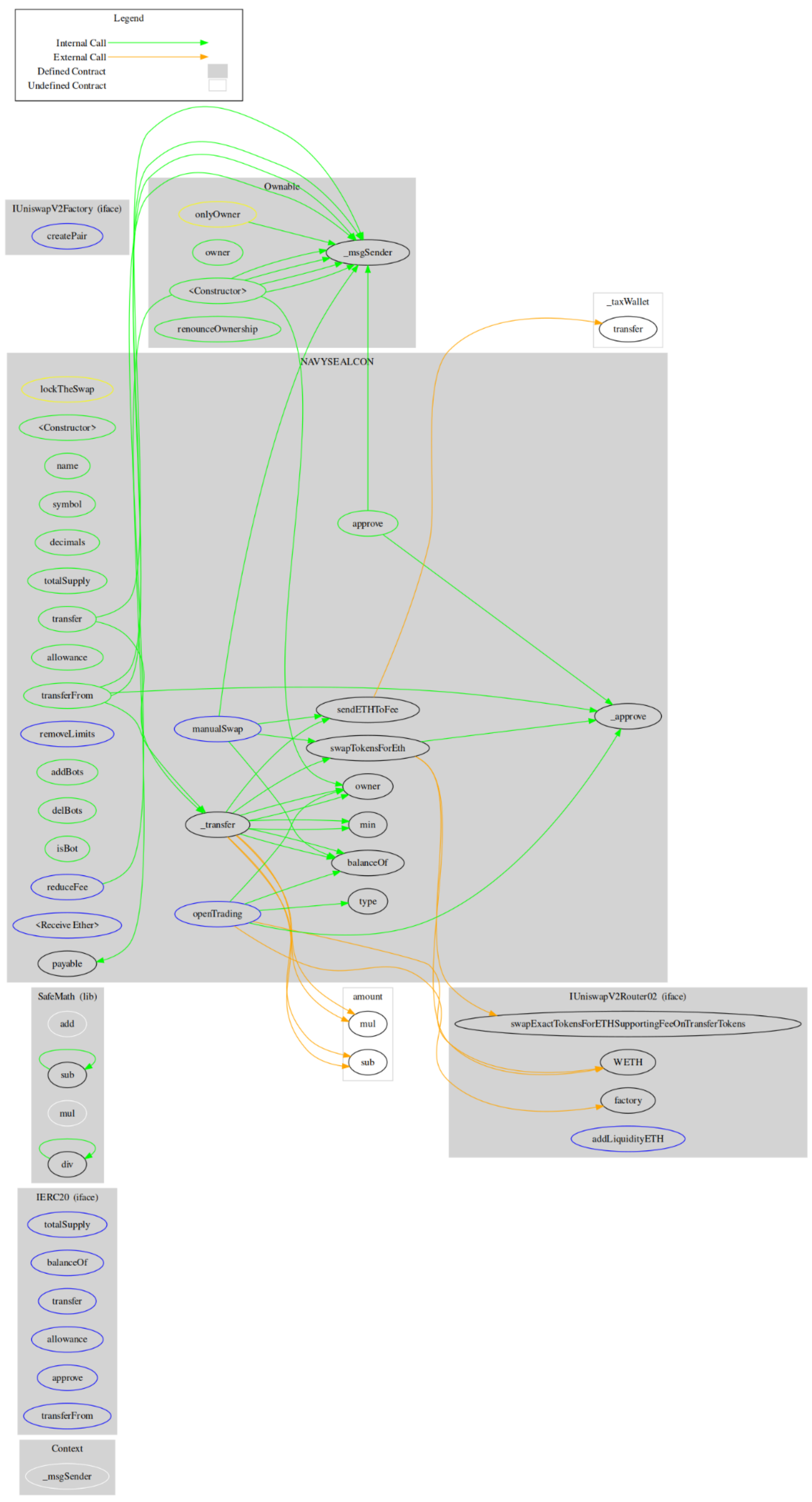
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
IUniswapV2Factory	Interface			
	createPair	External	✓	-
IUniswapV2Router02	Interface			
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
NAVYSEALCON	Implementation	Context, IERC20, Ownable		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-

	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_approve	Private	✓	
	_transfer	Private	✓	
	min	Private		
	swapTokensForEth	Private	✓	lockTheSwap
	removeLimits	External	✓	onlyOwner
	sendETHToFee	Private	✓	
	addBots	Public	✓	onlyOwner
	delBots	Public	✓	onlyOwner
	isBot	Public		-
	openTrading	External	✓	onlyOwner
	reduceFee	External	✓	-
		External	Payable	-
	manualSwap	External	✓	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IUniswapV2Factory (iface)
createPair
Ownable
onlyOwner
owner
_msgSender
<Constructor>
renounceOwnership
_taxWallet
transfer
NAVYSEALCON
lockTheSwap
<Constructor>
name
symbol
decimals
totalSupply
transfer
allowance
transferFrom
removeLimits
addBots
delBots
isBot
reduceFee
<Receive Ether>
payable
approve
sendETHToFee
_approve
manualSwap
swapTokensForEth
owner
_transfer
min
balanceOf
type
openTrading
SafeMath (lib)
add
sub
mul
div
amount
mul
sub
IUniswapV2Router02 (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
factory
addLiquidityETH
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender

Summary

Navy Seal contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Navyseal Token is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner has renounced ownership. There is also a limit of max 2% fees.

The contract's ownership has been renounced. The information regarding the transaction can be accessed through the following link:

https://etherscan.io/tx/0x37cd8742ad8ed58a3ee3703bcdf9eebe2c89263f2e67729c73ac46c52afcc337

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