Cyberscope

Audit Report

One80

February 2023

Type	ERC20
Network	ETH
Address	0xC1379686Efc619c96a0B995e1622ef3E5E023f7e
Audited by	© cyberscope

Table of Contents

Review

Contract Name	One80
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	200 runs
Explorer	https://etherscan.io/address/0xc1379686efc619c96a0b995e1622ef3e5e023f7e
Address	0xc1379686efc619c96a0b995e1622ef3e5e023f7e
Network	ETH
Symbol	One80
Decimals	18
Total Supply	200,000,000

Audit Updates

Initial Audit	22 Feb 2023

Source Files

Filename	SHA256
One80.sol	e34e21ddbf387d0b72af963881715e23b16f397d2be218c6c35f7f2ff6f6326f

Analysis

🔴 Critical 🟠 Medium ⚪ Minor / Informative 🔵 Pass

Severity	Code	Description	Status
⚪	ST	Stops Transactions	Unresolved
🔵	OCTD	Transfers Contract's Tokens	Passed
🔵	OTUT	Transfers User's Tokens	Passed
🔵	ELFM	Exceeds Fees Limit	Passed
🔵	ULTW	Transfers Liquidity to Team Wallet	Passed
🔴	MT	Mints Tokens	Unresolved
🔵	BT	Burns Tokens	Passed
🟠	BC	Blacklists Addresses	Unresolved

MT - Mints Tokens

Criticality	Critical
Location	One80.sol#L1177
Status	Unresolved

Description

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

```
function mint(address to, uint256 amount) public onlyRole(MINTER_ROLE) {
    _mint(to, 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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

ST - Stops Transactions

Criticality	Minor / Informative
Location	One80.sol#L1205
Status	Unresolved

Description

The contract `PAUSER` has the authority to stop the transactions for all users. The `PAUSER` may take advantage of it by calling the `pause()` function. As a result, all transactions will be paused.

```
function _beforeTokenTransfer(address from, address to, uint256 amount)
    internal
    whenNotPaused
    override
{
    super._beforeTokenTransfer(from, to, 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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

BC - Blacklists Addresses

Criticality	Medium
Location	One80.sol#L1181
Status	Unresolved

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `freeze()` function.

```
function freeze(address user) public  onlyRole(DEFAULT_ADMIN_ROLE){
     isFrozen[user] = true;
}
```

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	One80.sol#L1196
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.

```
address _token
```

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.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	One80.sol#L77,102,454,463
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function toString(uint256 value) internal pure returns (string memory) {
        // Inspired by OraclizeAPI's implementation - MIT licence
        //
https://github.com/oraclize/ethereum-api/blob/b42146b063c7d6ee1358846c198246239e9360
e8/oraclizeAPI_0.4.25.sol

        if (value == 0) {
            return "0";
...
        while (value != 0) {
            digits -= 1;
            buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
            value /= 10;
        }
        return string(buffer);
    }

...
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	One80.sol#L7,35,66,136,227,254,503,610,695,725,1114,1153
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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	One80.sol#L1200
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
erc20token.transfer(admin, balance)
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

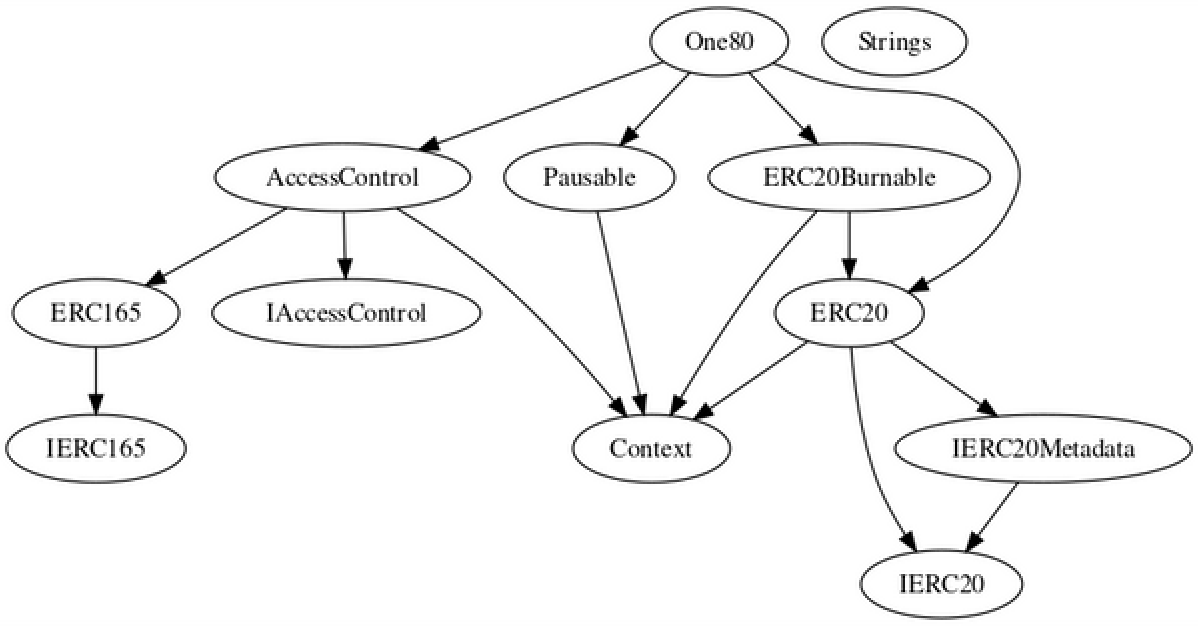
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC165	Interface			
	supportsInterface	External		-
ERC165	Implementation	IERC165		
	supportsInterface	Public		-
Strings	Library			
	toString	Internal		
	toHexString	Internal		
	toHexString	Internal		
IAccessControl	Interface			
	hasRole	External		-
	getRoleAdmin	External		-
	grantRole	External	✓	-
	revokeRole	External	✓	-
	renounceRole	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
AccessControl	Implementation	Context, IAccessCon		

		trol, ERC165		
	supportsInterface	Public		-
	hasRole	Public		-
	_checkRole	Internal		
	_checkRole	Internal		
	getRoleAdmin	Public		-
	grantRole	Public	✓	onlyRole
	revokeRole	Public	✓	onlyRole
	renounceRole	Public	✓	-
	_setupRole	Internal	✓	
	_setRoleAdmin	Internal	✓	
	_grantRole	Internal	✓	
	_revokeRole	Internal	✓	
Pausable	Implementation	Context		
		Public	✓	-
	paused	Public		-
	_requireNotPaused	Internal		
	_requirePaused	Internal		
	_pause	Internal	✓	whenNotPaused
	_unpause	Internal	✓	whenPaused
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-

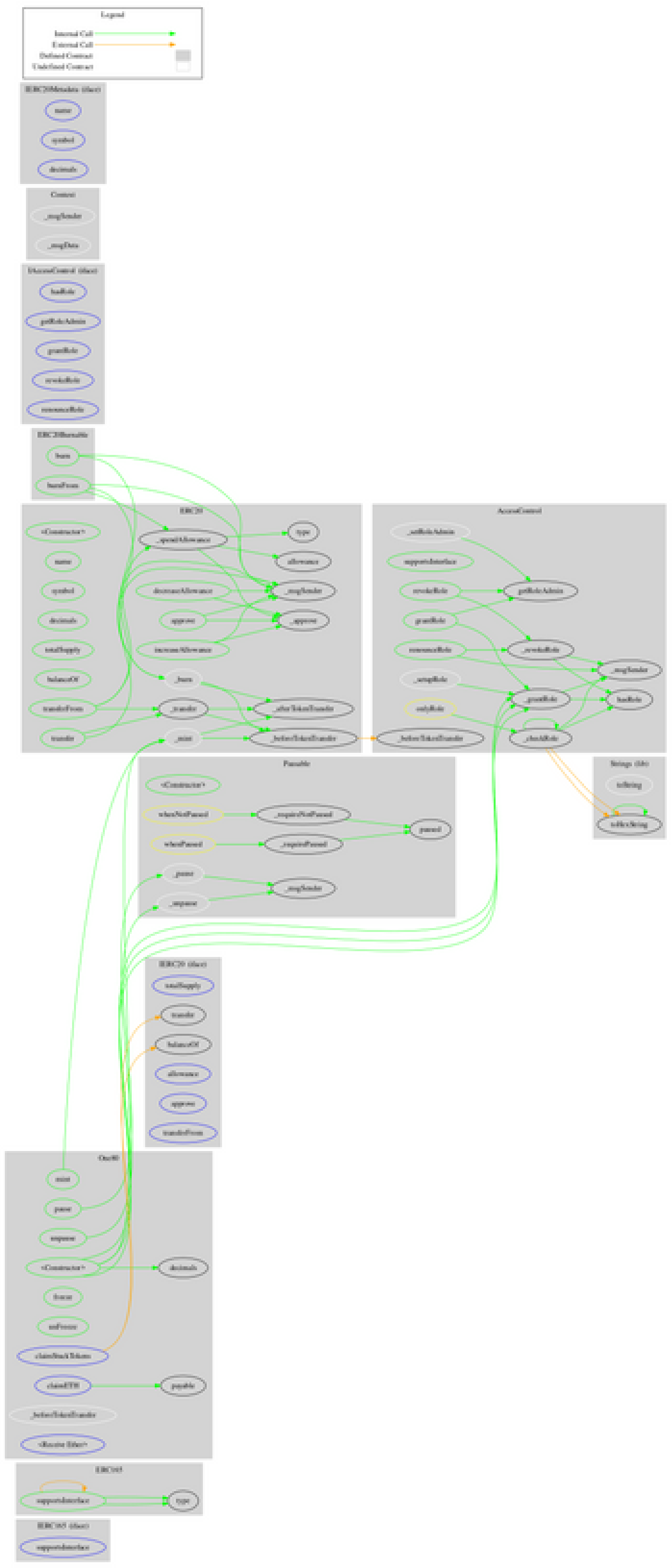
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
ERC20	Implementation	Context, IERC20, IERC20Metadata		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	

ERC20Burnable	Implementation	Context, ERC20		
	burn	Public	✓	-
	burnFrom	Public	✓	-
One80	Implementation	ERC20, ERC20Burnable, Pausable, AccessControl		
		Public	✓	ERC20
	pause	Public	✓	onlyRole
	unpause	Public	✓	onlyRole
	mint	Public	✓	onlyRole
	freeze	Public	✓	onlyRole
	unFreeze	Public	✓	onlyRole
	claimETH	External	✓	onlyRole
	claimStuckTokens	External	✓	onlyRole
	_beforeTokenTransfer	Internal	✓	whenNotPaused
		External	Payable	-

Inheritance Graph



Flow Graph



Summary

There are some functions that can be abused by the owner like stop transactions, mint tokens and blacklist addresses. 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.

Disclaimer

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