Cyberscope

Audit Report
Taveta

December 2023

Network	ETH
Address	0x0B29956d1A26caBc549a9538A97Dc8Abede4c28d
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	Unresolved
●	BC	Blacklists Addresses	Passed

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	TSM	Total Supply Misuse	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Review

Contract Name	Taveta
Compiler Version	v0.8.22+commit.4fc1097e
Optimization	200 runs
Explorer	https://etherscan.io/address/0x0b29956d1a26cabc549a9538a97dc8abede4c28d
Address	0x0b29956d1a26cabc549a9538a97dc8abede4c28d
Network	ETH
Symbol	TAV
Decimals	18

Audit Updates

Initial Audit	28 Dec 2023

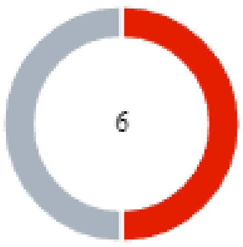
Source Files

Filename	SHA256
Taveta.sol	3ec03e0253484b31f9e99228c702ed9714 574ccf5f6f0f584001884efaf6b734
@openzeppelin/contracts/utils/Context.sol	9c1cc43aa4a2bde5c7dea0d4830cd42c54 813ff883e55c8d8f12e6189bf7f10a
@openzeppelin/contracts/utils/math/SafeMath.sol	fc16aa4564878e1bb65740239d0c142245 1cd32136306626ac37f5d5e0606a7b
@openzeppelin/contracts/token/ERC20/IERC20.sol	6f2faae462e286e24e091d7718575179644 dc60e79936ef0c92e2d1ab3ca3cee
@openzeppelin/contracts/token/ERC20/ERC20.sol	2d874da1c1478ed22a2d30dcf1a6ec0d09 a13f897ca680d55fb49fbcc0e0c5b1
@openzeppelin/contracts/token/ERC20/extensions /IERC20Metadata.sol	1d079c20a192a135308e99fa5515c27acfb b071e6cdb0913b13634e630865939
@openzeppelin/contracts/token/ERC20/extensions /ERC20Burnable.sol	2e6108a11184dd0caab3f3ef31bd15fed1b c7e4c781a55bc867ccedd8474565c
@openzeppelin/contracts/interfaces/draft-IERC609 3.sol	4aea87243e6de38804bf8737bf86f750443 d3b5e63dd0fd0b7ad92f77cdbd3e3
@openzeppelin/contracts/access/Ownable.sol	38578bd71c0a909840e67202db527cc6b4 e6b437e0f39f0c909da32c1e30cb81

Overview

The audit scope is to check for security vulnerabilities, validate the business logic and propose potential optimizations. The contract deviates significantly from fundamental ERC20 token creation coding principles. According to the previously mentioned issues, the contract cannot be assumed that it is in a production-ready state. Given these issues, it is not advisable to assume that the contract is in a production-ready state. The development team is strongly encouraged to re-evaluate the business logic and Solidity guidelines to ensure that the contract adheres to established best practices and security measures. The code's readability should also be improved by simplifying function definitions and using descriptive variable names, as this will enhance the contract's auditability and maintenance.

Findings Breakdown



●	Critical	3
●	Medium	0
●	Minor / Informative	3

Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	3	0	0	0
● Medium	0	0	0	0
● Minor / Informative	3	0	0	0

MT - Mints Tokens

Criticality	Critical
Location	Taveta.sol#L25
Status	Unresolved

Description

The `controller` role, which is assigned by the contract owner has the authority to mint tokens. The `controller` 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) external
onlyController {
  require((MAXSUP + amount) <= MAXIMUMSUPPLY, "Maximum supply
has been reached");
  _totalSupply = _totalSupply.add(amount);
  MAXSUP = MAXSUP.add(amount);
  _balances[to] = _balances[to].add(amount);
  _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.

Temporary Solutions, these measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

BT - Burns Tokens

Criticality	Critical
Location	Taveta.sol#L33
Status	Unresolved

Description

The `controller` role, which is assigned by the owner has the authority to burn tokens from a specific address. The `controller` may take advantage of it by calling the `burnFrom` function. As a result, the targeted address will lose the corresponding tokens.

In addition, the contract has redeclared the `_totalSupply` variable, which is separate from the `_totalSupply` in the inherited OpenZeppelin ERC20 contract. When the `burnFrom` function burns tokens, it correctly updates the original `_totalSupply` from the ERC20 contract. However, since the `totalSupply()` function in this contract returns the value of the redeclared `_totalSupply`, it does not accurately reflect the actual total supply after tokens are burned. This discrepancy leads to a situation where the reported total supply of tokens is higher than the actual supply.

```
function burnFrom(address account, uint256 amount) public
override {
  if (controllers[msg.sender]) {
    _burn(account, amount);
  } else {
    super.burnFrom(account, amount);
  }
}
```

Recommendation

The team is advised to reconsider the implementation of the contract and apply the necessary changes in order to make it functional.

They should also 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.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

TSM - Total Supply Misuse

Criticality	Critical
Location	Taveta.sol#L17,49
Status	Unresolved

Description

A significant issue with the management of the total supply in the contract is identified. The `_totalSupply` variable is redeclared, which is already defined and managed in the OpenZeppelin ERC20 base contract. Due to this redeclaration, the `_totalSupply` variable in Taveta shadows the one from the ERC20 implementation and is not updated correctly during token minting and burning operations.

For example, this issue is particularly evident after the contract is deployed and the constructor invokes `_mint(msg.sender, 100000000 * 10 ** 18)`. While this operation correctly increases the token balance of the deploying address, it fails to update the `_totalSupply`. Consequently, calls to the `totalSupply()` function, which returns the value of the redeclared `_totalSupply`, incorrectly report zero. This misalignment means that the actual token supply in circulation is not accurately reflected, leading to significant discrepancies and potential vulnerabilities in supply management.

This misrepresentation of the total token supply undermines the integrity of the token, potentially affecting trust and valuation. It can lead to confusion and misjudgment by users and external entities relying on the `totalSupply()` function for accurate information. Furthermore, operations like token minting and burning compound this issue, further diverging the reported supply from the actual supply.

```
uint256 private _totalSupply;
...
function totalSupply() public override view returns (uint256) {
    return _totalSupply;
}
```

Recommendation

It is advised for the team to conduct a comprehensive review of the contract's business logic and overall structure. to identify if the redeclaration of `_totalSupply` is necessary.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	Taveta.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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Taveta.sol#L18
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.

```
uint256 private MAXSUP
```

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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	Taveta.sol#L4
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 ^0.8.20;
```

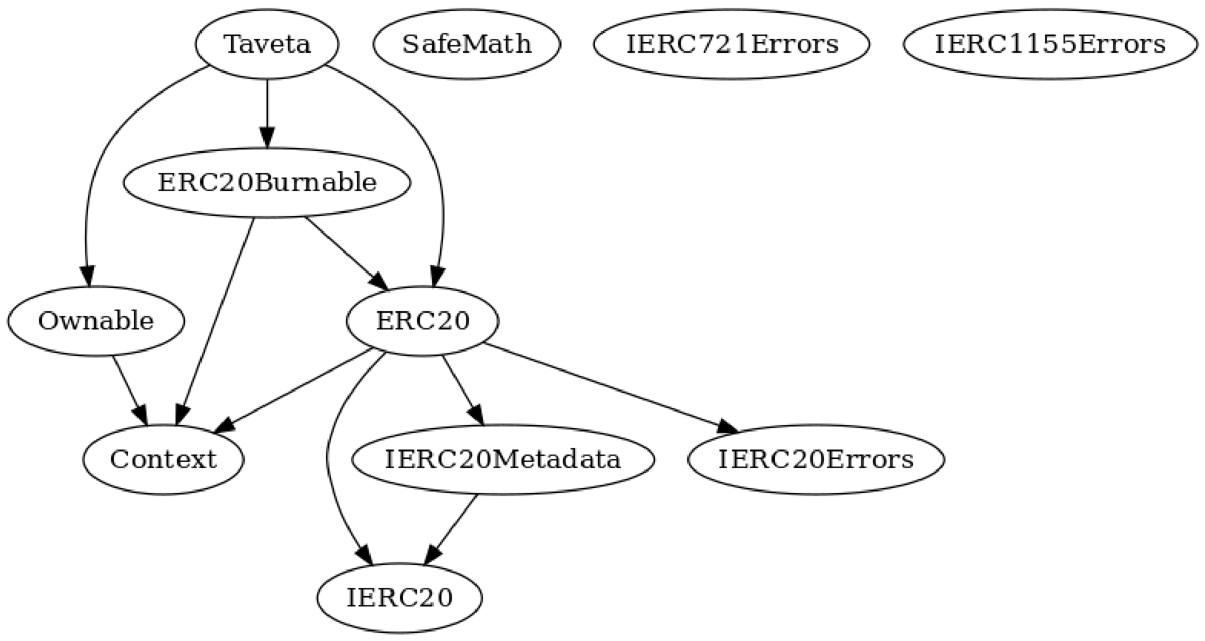
Recommendation

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.

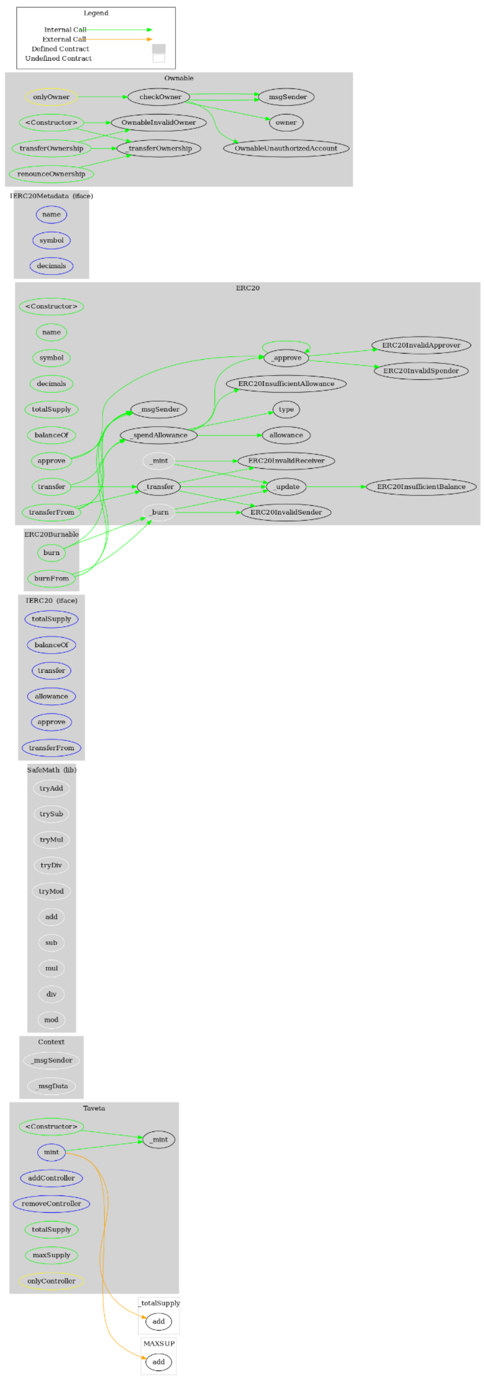
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Taveta	Implementation	ERC20, ERC20Burnable, Ownable		
		Public	✓	ERC20 Ownable
	mint	External	✓	onlyController
	burnFrom	Public	✓	-
	addController	External	✓	onlyOwner
	removeController	External	✓	onlyOwner
	totalSupply	Public		-
	maxSupply	Public		-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
onlyOwner
checkOwner
_msgSender
<Constructor>
OwnableInvalidOwner
owner
transferOwnership
_transferOwnership
OwnableUnauthorizedAccount
renounceOwnership
IERC20Metadata (iface)
name
symbol
decimals
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
approve
transfer
transferFrom
_approve
ERC20InvalidApprover
ERC20InvalidSpender
ERC20InsufficientAllowance
_msgSender
type
_spendAllowance
allowance
_mint
ERC20InvalidReceiver
_transfer
_update
ERC20InsufficientBalance
_burn
ERC20InvalidSender
ERC20Burnable
burn
burnFrom
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod
Context
_msgSender
_msgData
Taveta
<Constructor>
_mint
mint
addController
removeController
totalSupply
maxSupply
onlyController
_totalSupply
add
MAXSUP
add

Summary

Taveta contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. At this point, the contract is not in a production-ready state. There are some functions that can be abused by the owner like mint tokens and burn tokens from any address. if the contract owner abuses the mint functionality, then the contract will be highly inflated. if the contract owner abuses the burn functionality, then the users could lost their tokens.

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