Cyberscope

Audit Report

Ape Universe

February 2022

Type	ERC20
Network	AVAX
Address	0x6b0d2a3c37d551963275bB104F045F6a68AB6374
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	ApeUniverse
Compiler Version	v0.8.11+commit.d7f03943
Optimization	200 runs
Licence	None
Explorer	https://snowtrace.io/token/0x6b0d2a3c37d551963275bB104F045F6a68AB6374
Symbol	Verny Doge
Decimals	18
Total Supply	25,154,585,443.697758
Source	contract.sol
Domain	https://apes.money/

Audit Updates

Initial Audit	2nd March 2022
Corrected	

Contract Analysis

Critical Medium Minor Pass

Severity	Code	Description
●	ST	Contract Owner is not able to stop or pause transactions
●	OCTD	Contract Owner is not able to transfer tokens from specific address
●	OTUT	Owner Transfer User's Tokens
●	ELFM	Contract Owner is not able to increase fees more than a reasonable percent (25%)
●	ULTW	Contract Owner is not able to increase the amount of liquidity taken by dev wallet more than a reasonable percent
●	MT	Contract Owner is not able to mint new tokens
●	BT	Contract Owner is not able to burn tokens from specific wallet
●	BC	Contract Owner is not able to blacklist wallets from selling

ST - Stop Transactions

Criticality	critical
Location	contract.sol#L110

Description

The contract owner has the authority to stop selling transactions for all users. The owner may take advantage of it by setting the `sellFee` to a high value like 100. This behaviour will make the contract operate like a honeypot.

```
    if(to == liquidityPoolManager.getPair()) {
            fees = amount.mul(sellFee).div(100);
            amount = amount.sub(fees);
            address treasuryAddress = liquidityPoolManager.getTreasuryAddress();
            super._transfer(from, treasuryAddress, fees);
```

Recommendation

The contract could embody a check for not allowing setting the sellFee less than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply.

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.

OCTD - Owner Contract Tokens Drain

Criticality	minor
Location	OwnerRecovery.sol#L7

Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the `recoverLostAVAX` or `recoverLostTokens` functions.

```
    function recoverLostAVAX() external onlyOwner {
        payable(owner()).transfer(address(this).balance);
    }

    function recoverLostTokens(
        address _token,
        address _to,
        uint256 _amount
    ) external onlyOwner {
        IERC20(_token).transfer(_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.

ELFM - Exceed Limit Fees Manipulation

Criticality	critical
Location	contract.sol#L161,166

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `setTransferFee` function with a high percentage value.

```
function setTransferFee(uint256 transferFee_) external onlyOwner {
      transferFee = transferFee_;
      emit SetTransferFee(transferFee_);
  }
```

Recommendation

The contract could embody a check for the maximum acceptable value.

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.

MT - Mint Tokens

Criticality	critical
Location	contract.sol#L1,L892

Description

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

```
function accountReward(address account, uint256 amount)
        external
        onlyPlanetsManager
    {
        require(
            address(liquidityPoolManager) != account,
            "ApeUniverse: Use liquidityReward to reward liquidity"
        );
        super._mint(account, amount);
    }
```

Recommendation

The owner should carefully manage the credentials of the owner's account. We advised considering an extra-strong security mechanism that the actions may be quarantined by many users instead of one. The owner could also renounce the contract ownership for a period of time or pass the access to the zero address.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
	L01	Public Function could be Declared External
	L05	Unused State Variable
	L04	Conformance to Solidity Naming Conventions
	L09	Dead Code Elimination

L01 - Public Function could be Declared External

Criticality	minor
Location	/Ownable.sol#L54,62 /ERC20Burnable.sol#L20,35 /LiquidityPoolManagerImplementationPointer.sol#L30,34

Description

Public functions that are never called by the contract should be declared external to save gas.

```
changeWalletObserverImplementation
getWalletObserverImplementation
changeLiquidityPoolManagerImplementation
getLiquidityPoolManagerImplementation
burnFrom
burn
decreaseAllowance
increaseAllowance
transferFrom
...
```

Recommendation

Use the external attribute for functions never called from the contract.

L05 - Unused State Variable

Criticality	minor
Location	contracts/implementations/WalletObserverImplementationPointer.sol#L49

Description

There are segments that contain unused state variables.

```
__gap
```

Recommendation

Remove unused state variables.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contracts/helpers/OwnerRecovery.sol#L12,13,14 contracts/implementations/LiquidityPoolManagerImplementationPointer.sol#L49 contracts/implementations/WalletObserverImplementationPointer.sol#L49

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
__gap
_amount
_to
_token
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions

L09 - Dead Code Elimination

Criticality	minor
Location	/utils/Address.sol#L80,90,109,123,169,179,142,152,55,196 /utils/Context.sol#L21 /utils/math/SafeMath.sol#L93,191,151,217,168,22,64,76,47,35

Description

Functions that are not used in the contract, and make the code's size bigger.

```
trySub
tryMul
tryMod
tryDiv
tryAdd
sub
mod
div
add
...
```

Recommendation

Remove unused functions.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
ERC20	Implementation	Context, IERC20, IERC20Metadata		
	<Constructor>	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	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	

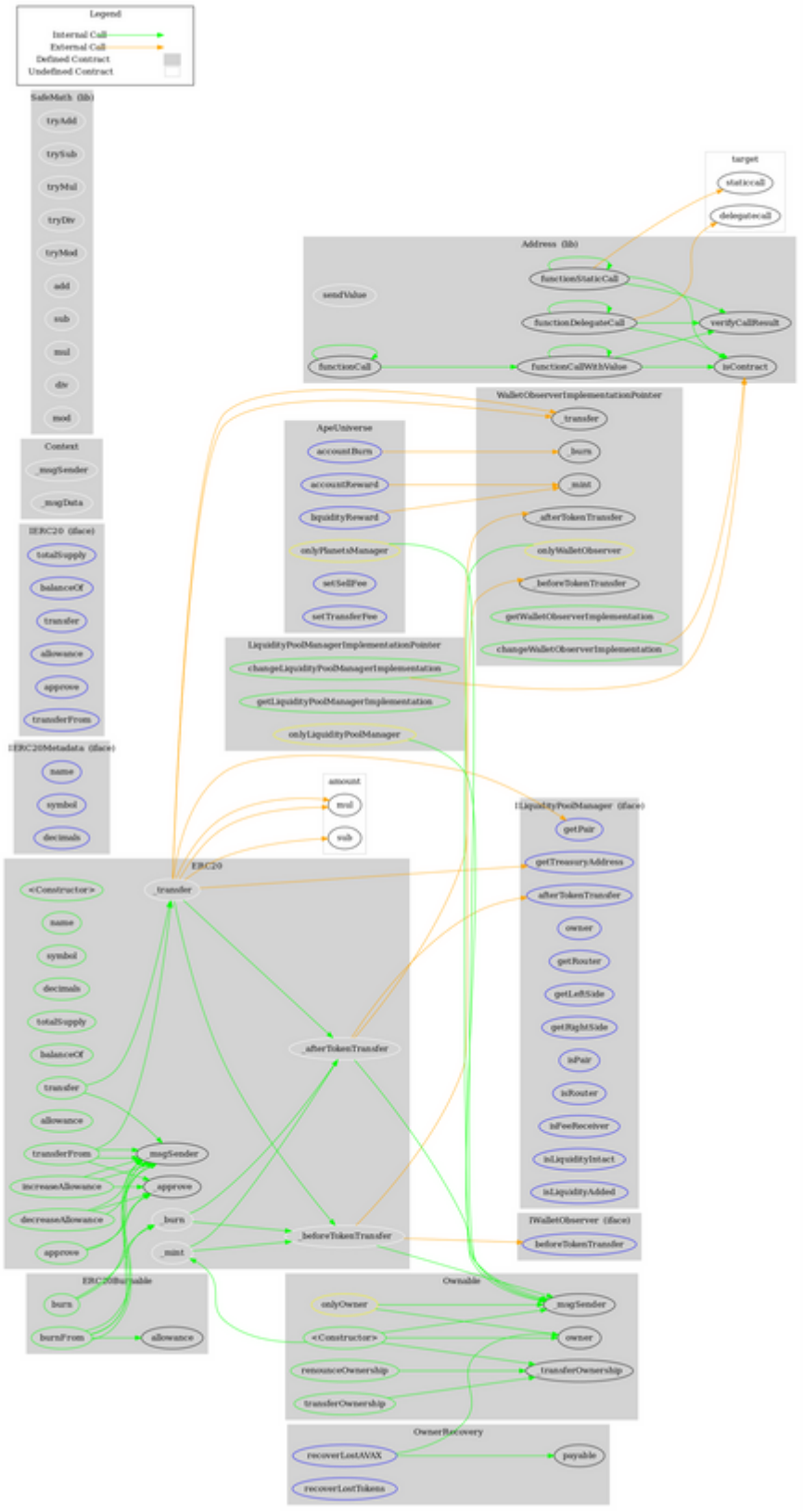
ERC20Burnable	Implementation	Context, ERC20		
	burn	Public	✓	-
	burnFrom	Public	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	verifyCallResult	Internal		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		

SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
ApeUniverse	Implementation	ERC20, ERC20Burnable, Ownable, OwnerRecovery, LiquidityPoolManagerImplementationPointer, WalletObserverImplementationPointer		
	<Constructor>	Public	✓	ERC20
	_beforeTokenTransfer	Internal	✓	
	_transfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
	accountBurn	External	✓	onlyPlanetsManager
	accountReward	External	✓	onlyPlanetsManager
	liquidityReward	External	✓	onlyPlanetsManager
	setSellFee	External	✓	onlyOwner
	setTransferFee	External	✓	onlyOwner

OwnerRecovery	Implementation	Ownable		
	recoverLostAVAX	External	✓	onlyOwner
	recoverLostTokens	External	✓	onlyOwner
LiquidityPoolManagerImplementationPointer	Implementation	Ownable		
	getLiquidityPoolManagerImplementation	Public		-
	changeLiquidityPoolManagerImplementation	Public	✓	onlyOwner
WalletObserverImplementationPointer	Implementation	Ownable		
	getWalletObserverImplementation	Public		-
	changeWalletObserverImplementation	Public	✓	onlyOwner
ILiquidityPoolManager	Interface			
	owner	External		-
	getRouter	External		-
	getPair	External		-
	getLeftSide	External		-
	getRightSide	External		-
	isPair	External		-
	isRouter	External		-
	isFeeReceiver	External		-
	isLiquidityIntact	External		-
	isLiquidityAdded	External		-
	afterTokenTransfer	External	✓	-
	getTreasuryAddress	External		-
IWalletObserver	Interface			

	beforeTokenTransfer	External	✓	-

Contract Flow



Domain Info

Domain Name	apes.money
Registry Domain ID	0e307ac83a954ab6bd6221ee42660401-DONUTS
Creation Date	2022-01-29T20:57:38Z
Updated Date	2022-02-04T20:36:18Z
Registry Expiry Date	2023-01-29T20:57:38Z
Registrar WHOIS Server	whois.godaddy.com/
Registrar URL	http://www.godaddy.com/domains/search.aspx?ci=8990
Registrar	GoDaddy.com, LLC
Registrar IANA ID	146

The domain has been created about 1 month before the creation of the audit. It will expire in 11 months.

There is no public billing information, the creator is protected by the privacy settings.

Summary

Ape Universe Token is an interesting project with a friendly and growing community. There are some functions that can be abused by the owner, like manipulating fees up to 100%, minting new tokens and transferring funds to the team’s wallet. The contract Owner can also convert the contract into a honeypot and disable all selling actions if he abuses the owner functions. 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

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment.

Cyberscope team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed.

The Cyberscope team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Cyberscope receive a payment to manipulate those results or change the awarding badge that we will be adding in our website.

Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token.

The Cyberscope team disclaims any liability for the resulting losses.

About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Cyberscope and Coinscope are aiming to make crypto discoverable and efficient globally. They provides all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io