Cyberscope

Audit Report

Digits DAO

December 2022

SHA256 248d43b885d50abb8b1fcb7469de43f927093bfdd81183d512fcc9090dd095a7

3eb09ab1e0ec0cd635d4d635c2bdb934540c42b8ea11492db811bee23648f68d

bf1e055d259d23d822843f235b383f0b34563f94c95d8d0527b18949cacb9c34

Audited by © cyberscope

Table of Contents

Contract Review

Contract Names	Testing Deploy
Digits	https://testnet.bscscan.com/address/0x5A78E0c801929a8C27C51dF6641257B29A11e342
DividendTracker	https://testnet.bscscan.com/address/0xd8AC8abEe3781B0459C0F2c9c8f905A0FA9Fed8d
TokenStorage	https://testnet.bscscan.com/address/0x4ede2995804251390cbb237Fb7a843F03b41D7a5

Audit Updates

Initial Audit	19 Dec 2022 https://github.com/cyberscope-io/audits/blob/main/digits-dao/v1/audit.pdf
Corrected Phase 2	30 Dec 2022

Source Files

Filename	SHA256
Digits.sol	248d43b885d50abb8b1fcb7469de43f927093bfdd81183d512fcc9090dd095a7
DividendTracker.sol	3eb09ab1e0ec0cd635d4d635c2bdb934540c42b8ea11492db811bee23648f68d
interfaces/IDigits.sol	66b6f984edfff8b497f5daeac74ff1fb67bae825ecc61b7f0616054ff37c47ea
interfaces/IDividendTracker.sol	233daa96bb68f398d75609faf9a50bda7562a83e18268119c62cd20f3757471a
interfaces/ITokenStorage.sol	aa4937c7971b362d7664cf35b292afe2fd43e8235079710d0765870deacddc96
TokenStorage.sol	bf1e055d259d23d822843f235b383f0b34563f94c95d8d0527b18949cacb9c34

Introduction

Digits DAO consists of four contracts:

- Digits
- DividendTracker
- TokenStorage
- MultiRewards

This audit report is referring to the first three contracts. The audit report for the `MultiRewards` contract can be found at https://github.com/cyberscope-io/audits/tree/main/digits-dao/MultiRewards.pdf.

Roles

Digits

The Digits contract has two roles, the USER role and the OWNER role.

The OWNER role has the authority to

- Include/Exclude an address from fees.
- Include/Exclude an address from the max tx amount.
- Include/Exclude an address from the max wallet amount
- Set fees up to 24% combined.
- Enable/Disable fees.
- Claim all the balance of the contract.
- Renounce/Transfer ownership.

The USER role has the authority to

- Make transactions.
- Claim dividends distributed as DAI tokens.

DividendTracker

The DividendTracker contract has two roles, the USER role, and the OWNER role.

The OWNER role has the authority to

- Renounce/Transfer ownership.
- Set the balance of an account.
- Include/Exclude an account from dividends.
- Transfer an account's dividends to that account.

The USER role has the authority to

- Transfer dividends to the contract.

TokenStorage

The DividendTracker contract has two roles, the MANAGER role, and the OWNER role.

The OWNER role has the authority to

- Renounce/Transfer ownership.
- Add/Remove managers.

The MANAGER role has the authority to

- Transfer DAI to an address.
- Swap tokens for DAI.
- Distribute dividends.

Contract Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	OCTD	Transfers Contract's Tokens	Acknowledged
●	PVC	Price Volatility Concern	Acknowledged
●	US	Untrusted Source	Acknowledged
●	L18	Multiple Pragma Directives	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Acknowledged
●	L13	Divide before Multiply Operation	Unresolved
●	L19	Stable Compiler Version	Unresolved

OCTD - Transfers Contract's Tokens

Criticality	Minor / Informative
Location	Digits.sol#L662
Status	Acknowledged

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 `rescueToken` function.

```
function rescueToken(address _token, uint256 _amount) external onlyOwner {
  IERC20(_token).transfer(msg.sender, _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.

Team Update

"*Owner will likely be a multisig, also we don't expect ERC20 tokens to be sent to Digits contract per se.*"

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	Digits.sol#L602
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapTokensAtAmount` sets a threshold where the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH. It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
function updateDividendSettings(bool _swapEnabled, uint256 _swapTokensAtAmount, bool
_swapAllToken) external onlyOwner {
  swapEnabled = _swapEnabled;
  swapTokensAtAmount = _swapTokensAtAmount;
  swapAllToken = _swapAllToken;

  emit UpdateDividendSettings(_swapEnabled, _swapTokensAtAmount, _swapAllToken);
}
```

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.

Team Update

“We are fine with this, it’s unlikely that token swap will trigger with great token amount, as each transfer that fees reach threshold they are swapped and we control the threshold.”

US - Untrusted Source

Criticality	Minor / Informative
Location	Digits.sol#L117,440
Status	Unresolved

Description

The contract uses an external contract in order to determine the transaction's flow. The external contract is untrusted. As a result, it may produce security issues and harm the transactions.

```
dividendTracker = new DividendTracker(dai, address(this), uniswapRouter);
...
function setTokenStorage(address _tokenStorage) external onlyOwner {
  ...
  tokenStorage = ITokenStorage(_tokenStorage);
}
```

Recommendation

The contract should use a trusted external source. A trusted source could be either a commonly recognized or an audited contract. The pointing addresses should not be able to change after the initialization.

Team Update

“DividendTracker contract is created by the Digits contract on deployment and is part of this audit. TokenStorage contract is settable only once.”

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	TokenStorage.sol#L3 DividendTracker.sol#L3 Digits.sol#L3,4
Status	Unresolved

Description

If the contract includes multiple conflicting pragma directives, it may produce unexpected errors. To avoid this, it's important to include the correct pragma directive at the top of the contract and to ensure that it is the only pragma directive included in the contract.

```
pragma solidity ^0.8.10;
pragma solidity ^0.8.10;

pragma solidity ^0.8.10;
pragma experimental ABIEncoderV2
```

Recommendation

It is important to include only one pragma directive at the top of the contract and to ensure that it accurately reflects the version of Solidity that the contract is written in. By including all required compiler options and flags in a single pragma directive, you can avoid conflicts and ensure that the contract can be compiled correctly.

Team Update

“*Each contract has it’s own pragma directive.*”

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	TokenStorage.sol#L55,60,129 DividendTracker.sol#L17,18,19,20 Digits.sol#L22,23,411,482,487,501,521,522,523,538,543,548,573,574,575,619,623
Status	Acknowledged

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 _address
address _liquidityWallet
string private constant _name = "Digits_DividendTracker"
string private constant _symbol = "Digits_DividendTracker"
uint256 private constant minTokenBalanceForDividends = 10000 * (10**18)
uint256 private constant magnitude = 2**128
string private constant _name = "Digits"
string private constant _symbol = "DIGITS"
address[] memory _users
address _multiRewards
address _tokenStorage
address _marketingWallet
uint256 _treasuryFee
uint256 _liquidityFee

...
```

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.

Team Update

“Code was just repurposed, we didn’t want to make too many changes to working code, so we will leave these as is.”

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	Digits.sol#L424,429,446,448
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
swapTokensDividends = (tokens * dividendFeeBPS) / totalFeeBPS
uint256 daiDividends = (daiSwapped * swapTokensDividends) /
swapTokensTotal
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	TokenStorage.sol#L3 interfaces/ITokenStorage.sol#L3 interfaces/IDividendTracker.sol#L3 interfaces/IDigits.sol#L3 DividendTracker.sol#L3 Digits.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 ^0.8.10;
```

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**
Digits	Implementation	Ownable, IERC20, IDigits		
		Public	✓	-
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	Public		-
	balanceOf	Public		-
	allowance	External		-
	approve	External	✓	-
	withdrawableDividendOf	External		-
	isExcludedFromDividends	External		-
	isExcludedFromFees	External		-
	isExcludedFromMaxTx	External		-
	isExcludedFromMaxWallet	External		-
	increaseAllowance	External	✓	-
	decreaseAllowance	External	✓	-

	triggerDividendDistribution	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	claim	External	✓	-
	compound	External	✓	-
	_transfer	Internal	✓	
	_executeTransfer	Private	✓	
	_approve	Private	✓	
	_mint	Private	✓	
	includeToWhiteList	Private	✓	
	_executeSwap	Private	✓	
	_setAutomatedMarketMakerPair	Private	✓	
	setMultiRewardsAddress	External	✓	onlyOwner
	setTokenStorage	External	✓	onlyOwner
	setWallet	External	✓	onlyOwner
	setAutomatedMarketMakerPair	External	✓	onlyOwner
	setFee	External	✓	onlyOwner
	setSwapEnabled	External	✓	onlyOwner
	setTaxEnabled	External	✓	onlyOwner
	setCompoundingEnabled	External	✓	onlyOwner
	setMaxTxBPS	External	✓	onlyOwner
	setMaxWalletBPS	External	✓	onlyOwner

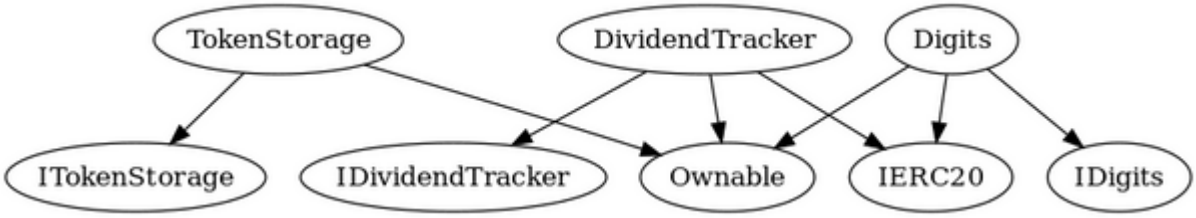
	openTrading	External	✓	onlyOwner
	updateDividendSettings	External	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	excludeFromDividends	External	✓	onlyOwner
	excludeFromMaxTx	Public	✓	onlyOwner
	excludeFromMaxWallet	Public	✓	onlyOwner
	rescueToken	External	✓	onlyOwner
	rescueETH	External	✓	onlyOwner
DividendTracker	Implementation	Ownable, IERC20, IDividendTracker		
		Public	✓	-
	distributeDividends	External	✓	-
	setBalance	External	✓	onlyOwner
	excludeFromDividends	External	✓	onlyOwner
	processAccount	External	✓	onlyOwner
	compoundAccount	External	✓	onlyOwner
	isExcludedFromDividends	External		-
	withdrawableDividendOf	Public		-
	withdrawnDividendOf	Public		-
	accumulativeDividendOf	Public		-
	getAccountInfo	External		-

	getLastClaimTime	External		-
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public		-
	allowance	Public		-
	approve	Public		-
	transferFrom	Public		-
	_setBalance	Internal	✓	
	_mint	Private	✓	
	_burn	Private	✓	
	_withdrawDividendOfUser	Private	✓	
	_compoundDividendOfUser	Private	✓	
IDigits	Interface			
	claim	External	✓	-
	triggerDividendDistribution	External	✓	-
IDividendTracker	Interface			

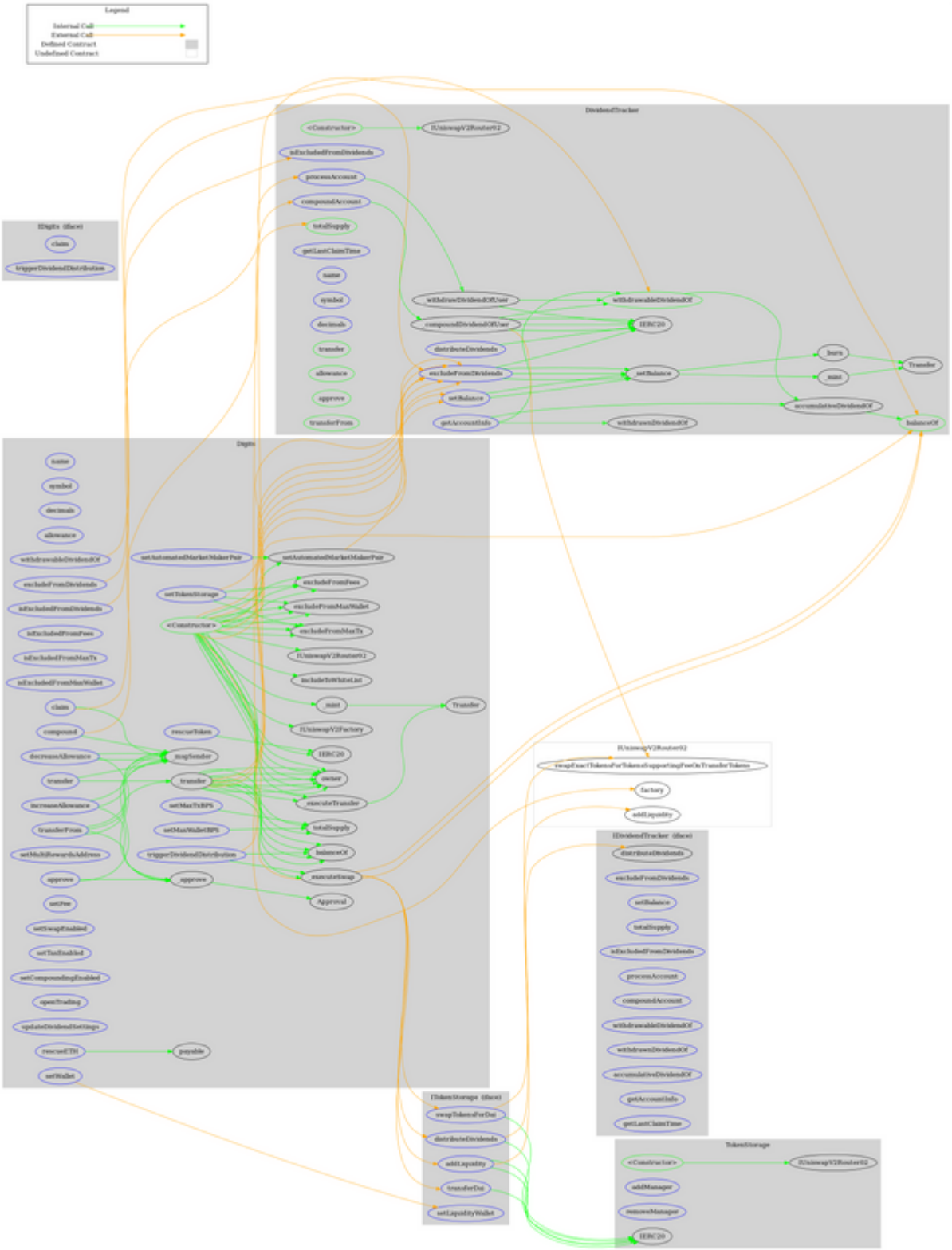
	distributeDividends	External	✓	-
	excludeFromDividends	External	✓	-
	setBalance	External	✓	-
	totalSupply	External		-
	isExcludedFromDividends	External		-
	processAccount	External	✓	-
	compoundAccount	External	✓	-
	withdrawableDividendOf	External		-
	withdrawnDividendOf	External		-
	accumulativeDividendOf	External		-
	getAccountInfo	External		-
	getLastClaimTime	External		-
ITokenStorage	Interface			
	swapTokensForDai	External	✓	-
	transferDai	External	✓	-
	addLiquidity	External	✓	-
	distributeDividends	External	✓	-
	setLiquidityWallet	External	✓	-
TokenStorage	Implementation	Ownable, ITokenStorage		

		Public	✓	-
	addManager	External	✓	onlyOwner
	removeManager	External	✓	onlyOwner
	transferDai	External	✓	-
	swapTokensForDai	External	✓	-
	addLiquidity	External	✓	-
	distributeDividends	External	✓	-
	setLiquidityWallet	External	✓	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
DividendTracker
IDigits (iface)
Digits
IUniswapV2Router02
IDividendTracker (iface)
ITokenStorage (iface)
TokenStorage

Summary

Digits DAO contracts implement a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements.

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