PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Digits DAO

07 February 2022

paladinsec.co

info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Paladin is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. Paladin may, at its discretion, claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for Digits DAO on the Avalanche network. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	Digits DAO
URL	https://digitsdao.finance/
Platform	Avalanche
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
`Digits`	`0x18e2269f98db2eda3cfc06e6cca384b291e553d9`	MATCH
`TokenStorage`	`0x94de014e5b5221a9e1b7c02518463074a79dde6a`	MATCH
`DividendTracker`	`0x6d39663ec3b905c019642e75fe674fc67384e58e`	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	7	7	-	-
Medium	6	5	-	1
Low	2	2	-	-
Informational	15	15	-	-
Total	**30**	**29**	**-**	**1**

Classification of Issues

Severity	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 Digits

ID	Severity	Summary	Status
01	HIGH	Gov Privilege: Token owner can drain all DAI dividends	RESOLVED
02	HIGH	Gov Privilege: `setTokenStorage` could be used to revert transactions, siphon fees or turn the token into a honeypot	RESOLVED
03	HIGH	Gov Privilege: `updateUniswapV2Router` could be used to revert transactions, siphon fees or turn the token into a honeypot	RESOLVED
04	HIGH	Gov Privilege: Fees are freely adjustable up to over 100%	RESOLVED
05	HIGH	Gov Privilege: Owner can blacklist wallets	RESOLVED
06	HIGH	`uniswapRouter` is initially set to an invalid address	RESOLVED
07	MEDIUM	Gov Privilege: Setting the liquidity wallet to the zero address turns the token into a honeypot	RESOLVED
08	MEDIUM	Gov Privilege: All tokens are pre-minted to the contract owner	RESOLVED
09	MEDIUM	`DEAD` address are not excluded from dividends	RESOLVED
10	LOW	`automatedMarketMakerPairs` may get out of sync with `uniswapV2Pair`	RESOLVED
11	LOW	`maxTxBPS` is initially set below the minimum enforced value	RESOLVED
12	INFO	Unnecessary zero address checks in `_executeTransfer`	RESOLVED
13	INFO	A number of variables are unused	RESOLVED
14	INFO	dividendTracker should be made immutable	RESOLVED
15	INFO	`_name`, `_symbol`, `dai` and `uniswapRouter` can be made `constant`	RESOLVED
16	INFO	A number of variables can be made external	RESOLVED
17	INFO	Lack of events for functions that change sensitive variables	RESOLVED
18	INFO	A number of return values are not handled	RESOLVED

1.3.2 TokenStorage

ID	Severity	Summary	Status
19	HIGH	Gov Privilege: Owner can add any address as an authorized manager to withdraw funds or turn the Digits token into a honeypot	RESOLVED
20	MEDIUM	Gov Privilege: Owner can update token address, turning the Digits token into a honeypot	RESOLVED
21	MEDIUM	LP tokens from addLiquidity are sent to `liquidityWallet`	ACKNOWLEDGED
22	MEDIUM	`setLiquidityWallet` and `updateUniswapV2Router` lack zero address checks	RESOLVED
23	INFO	`success` is unused in `distributeDividends`	RESOLVED
24	INFO	`dividendTracker` should be made `immutable`	RESOLVED
25	INFO	`dai` can be made constant	RESOLVED
26	INFO	A number of variables can be made external	RESOLVED

1.3.3 DividendTracker

ID	Severity	Summary	Status
27	INFO	Ambiguous error in `distributeDividends`	RESOLVED
28	INFO	`tokenAddress` should be made `immutable`	RESOLVED
29	INFO	`dai`, `_name` and `_symbol` can be made constant	RESOLVED
30	INFO	A number of variables can be made external	RESOLVED

2 Findings

2.1 Digits

Digits (DIGITS) is a dividend distributing token that has fees on transfer. Aside from allowing users to claim dividends (distributed as DAI tokens), it also allows them to compound these dividends into further Digits tokens.

For each non-whitelisted transfer, a fee of 12% is enforced: 3% reflection to dividends, 2% for liquidity and 7% for marketing. The fee percentages can be changed by the token owner, up to a maximum of 24% total.

These fees are transferred to a TokenStorage contract, and whenever the balance reaches a certain threshold (initially 100,000 Digits tokens), they will be swapped on SushiSwap for DAI and the dividends portion transferred to a DividendTracker contract, ready for distribution.

2.1.1 Token Overview

Address	`0x18e2269f98db2eda3cfc06e6cca384b291e553d9`
Token Supply	1,000,000,000
Decimal Places	18
Transfer Max Size	0.49% of the total supply
Transfer Min Size	None
Max Wallet Size	2% of the total supply
Transfer Fees	12%, can be set to a maximum of 24%
Pre-mints	1,000,000,000

2.1.2 Privileges

The following functions can be called by the owner of the contract:

- `renounceOwnership`
- `transferOwnership`
- `openTrading`
- `excludeFromFees`
- `excludeFromDividends`
- `setWallet`
- `setAutomatedMarketMakerPair`
- `setFee`
- `setSwapEnabled`
- `setTaxEnabled`
- `setCompoundingEnabled`
- `updateDividendSettings`
- `setMaxTxBPS`
- `excludeFromMaxTx`
- `setMaxWalletBPS`
- `excludeFromMaxWallet`

2.1.3 Issues & Recommendations

Issue #01	Gov Privilege: Token owner can drain all DAI dividends
Severity	HIGH SEVERITY
Description	The token owner can call `manualSendDividend` on the token contract to transfer any or all of the DAI dividends accumulated on the DividendTracker to any address. Users would then be unable to claim their dividends.
Recommendation	Consider removing this function. If the intent is to be able to withdraw any excess DAI accumulated on the DividendTracker, then consider modifying the function to ensure enough DAI remains for all users to be able to claim their dividends.
Resolution	RESOLVED This function has been removed.

Issue #02	Gov Privilege: `setTokenStorage` could be used to revert transactions, siphon fees or turn the token into a honeypot
Severity	HIGH SEVERITY
Description	The owner can change the TokenStorage contract used to store transfer fees, even after it has initially been set (including to the zero address). This contract could be a malicious contract that simply keeps the tokens sent to it and thus siphons all the fees. Additionally, this contract could be used to revert sell transactions turning the token into a honeypot.
Recommendation	Consider removing this function and then creating the TokenStorage contract immutably in the constructor. If it is still desirable to keep the function, then consider only allowing `tokenStorage` to be initialized once, and disallowing subsequent updates. Also consider requiring the `tokenStorage` address to be non-zero when it is set.
Resolution	RESOLVED This function can only be called once to initialize the TokenStorage contract.

Issue #03	**Gov Privilege: `updateUniswapV2Router` could be used to revert transactions, siphon fees or turn the token into a honeypot**
Severity	HIGH SEVERITY
Description	The owner can update the router that generates liquidity to an address or contract of choice (including the zero address). This contract could be a malicious contract that simply keeps the tokens sent to it and thus siphons all the fees. Additionally, this contract could be used to revert sell transactions turning the token into a honeypot.
Recommendation	Consider removing this function. If this is not possible, consider using an Owner account that is behind a significantly long timelock so investors can reasonably see this change coming and inspect the new router. Also consider requiring the router address to be non-zero.
Resolution	RESOLVED This function has been removed.

Issue #04	**Gov Privilege: Fees are freely adjustable up to over 100%**
Severity	HIGH SEVERITY
Description	The owner of the contract can set the individual fees to any variable. This might deter investors as they could be wary that these fees might one day be set to 100% to force transfers to go to the contract owner.
Recommendation	Consider adding an explicit cap to the total fee on every fee adjustment function.
Resolution	RESOLVED Each tax component can be no more than 8%, giving a total maximum cap of 24%.

Issue #05	Gov Privilege: Owner can blacklist wallets
Severity	HIGH SEVERITY
Description	Currently, the owner can blacklist wallets from purchasing or selling the tokens they hold. This might deter investors from purchasing the token.
Recommendation	Consider either removing the blacklist functionality, or only allowing it to be called for a certain duration after liquidity has been added. Alternatively, consider putting the token behind a significantly long timelock so investors can see any change coming.
Resolution	RESOLVED This functionality has been removed.

Issue #06	`uniswapRouter` is initially set to an invalid address
Severity	HIGH SEVERITY
Location	Line 27 `address public uniswapRouter = address(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D);  // sushi router`
Description	The address `uniswapRouter` is initially set to is just a user wallet on the Avalanche chain, and does not correspond to the comment of it being the SushiSwap router. As this will cause an error when used in the constructor, it is unlikely this contract will be deployable on mainnet.
Recommendation	Consider setting the initial value of `uniswapRouter` to a valid router address.
Resolution	RESOLVED The address has been updated to the correct SushiSwap router address.

Issue #07	Gov Privilege: Setting the liquidity wallet to the zero address turns the token into a honeypot
Severity	MEDIUM SEVERITY
Description	The owner can set the liquidity wallet address to zero, however, liquidity cannot be added to the zero address. Thus, if the governance ever sets the liquidity wallet to the zero address, they would effectively block all sell transactions, turning the token into a honeypot since `_executeSwap` is not called on purchases.
Recommendation	Consider requiring the liquidity wallet to be non-zero in both the constructor and `setWallet`. `require(_liquidityWallet != address(0), "zero!");`
Resolution	RESOLVED

Issue #08	Gov Privilege: All tokens are pre-minted to the contract owner
Severity	MEDIUM SEVERITY
Description	The contract owner may have many legitimate uses for the pre-minted tokens including, but not limited to, the injection of initial liquidity, token presale, or airdrops; however, they may also use these pre-minted tokens for dumping after adding liquidity.
Recommendation	Consider being forthright by letting your community know how much was minted, where they are currently stored, if a vesting contract was used for token unlocking, and finally the purpose of any remaining held tokens.
Resolution	RESOLVED The project has documented to its users the total supply and how it will be distributed: 15% to the team, 10% to treasury, 75% as liquidity (which will be locked with RugDoc).

Issue #09	**DEAD address is not excluded from dividends**
Severity	MEDIUM SEVERITY
Description	The `DEAD` address is not excluded from dividends, so any burned tokens will still be sharing dividends, which will accumulate on the DividendTracker. These can then be withdrawn by the contract owner via `manualSendDividend`.
Recommendation	Exclude the `DEAD` address from dividends using `dividendTracker.excludeFromDividends`.
Resolution	RESOLVED This is now excluded in the constructor.

Issue #10	**`automatedMarketMakerPairs` may get out of sync with `uniswapV2Pair`**
Severity	LOW SEVERITY
Description	Currently, the token owner needs to take care to manually call `setAutomatedMarketMakerPair` whenever they call `updateUniswapV2Router`. If they forget to do this, then the `automatedMarketMakerPairs` mapping will fail to contain the updated liquidity pair address.
Recommendation	Add a call to `_setAutomatedMarketMakerPair` from `updateUniswapV2Router`.
Resolution	RESOLVED As `updateUniswapV2Router` has been removed, this issue is no longer relevant.

Issue #11	maxTxBPS is initially set below the minimum enforced value
Severity	LOW SEVERITY
Description	When the contract is deployed, the antiwhale variable `maxTxBPS` is set to 49 (0.49%). However, in the `setMaxTxBPS` function that sets and enforces the bounds of this variable, the minimum value is 75 (0.75%). Users reading this function code may assume this value is the minimum the variable can be set to, when in fact it can be lower.
Recommendation	Consider setting the initial value of `maxTxBPS` to 75 to reduce potential confusion.
Resolution	RESOLVED `setMaxTxBPS` now has a minimum value of 0.49% to match the constructor.

Issue #12	Unnecessary zero address checks in `_executeTransfer`
Severity	INFORMATIONAL

Location	Line 1579~
	`require(sender != address(0), "Digits: transfer from the zero address");`
	`require(recipient != address(0), "Digits: transfer to the zero address");`

Description	`sender` and `recipient` are already guaranteed to be non-zero by the calling function `_transfer`.
Recommendation	Consider removing these checks.
Resolution	RESOLVED These checks have been removed.

Issue #13	A number of variables are unused
Severity	INFORMATIONAL
Description	Variables defined in a contract but not used within said contract could confuse third-party auditors. They furthermore increase the contract length and bytecode size for no reason: - `event BlacklistEnabled` (never emitted) - `_burn` (private, not called) - `liquidityWallet` (set but never read)
Recommendation	Consider removing the above variables to keep the contract short and simple.
Resolution	RESOLVED These variables have been removed.

Issue #14	`dividendTracker` should be made `immutable`
Severity	INFORMATIONAL
Description	Variables that are defined within the constructor but further remain unchanged should be marked as immutable to save gas and to ease the reviewing process of third-parties.
Recommendation	Consider marking this variable as immutable.
Resolution	RESOLVED

Issue #15	`_name`, `_symbol`, `dai` and `uniswapRouter` can be made `constant`
Severity	INFORMATIONAL
Description	Variables that are never modified can be indicated as such with the constant keyword. This is considered best practice since it makes the code more accessible for third-party reviewers and saves gas.
Recommendation	Consider making the above variables explicitly constant.
Resolution	RESOLVED

Issue #16	A number of variables can be made external
Severity	INFORMATIONAL
Description	Functions that are not used within the contract but only called externally can be indicated as such with the external keyword. This can in certain cases reduce gas and can ease the reviewing process of third parties: - `name` - `symbol` - `decimals` - `allowance` - `approve` - `increaseAllowance` - `decreaseAllowance` - `transfer` - `transferFrom` - `setTokenStorage` - `isExcludedFromFees` - `excludeFromDividends` - `isExcludedFromDividends` - `setAutomatedMarketMakerPair` - `updateUniswapV2Router` - `claim` - `compound` - `withdrawableDividendOf` - `withdrawnDividendOf` - `accumulativeDividendOf` - `getAccountInfo` - `getLastClaimTime` - `isExcludedFromMaxTx` - `isExcludedFromMaxWallet` - `blackList` - `removeFromBlacklist` - `blackListMany` - `unBlackListMany`
Recommendation	Consider marking the above functions as external.
Resolution	RESOLVED

Issue #17	Lack of events for functions that change sensitive variables
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables such as whitelists and fee percentages should emit events: - `setFee` - `setTokenStorage` - `updateDividendSettings` - `setMaxTxBPS` - `excludeFromMaxTx` - `setMaxWalletBPS` - `excludeFromMaxWallet` - `blackList` - `removeFromBlacklist`
Recommendation	Add events for the setter functions which modify sensitive state variables.
Resolution	RESOLVED

Issue #18	A number of return values are not handled
Severity	INFORMATIONAL
Description	The return values of the following function calls are not handled: - `uniswapV2Router.addLiquidity` - `dividendTracker.processAccount` - `dividendTracker.compoundAccount`
Recommendation	Consider using variables to receive the return value of the above functions and handle both success and failure cases if needed by the business logic
Resolution	RESOLVED

2.2 TokenStorage

The token storage contract receives the taxable portion of Digits transfers and has functions for swapping these to DAI and adding Digits/DAI liquidity.

2.2.1 Privileged Roles

The following functions can be called only by the owner of the contract:

- `addManager`
- `removeManager`

The following functions can be called by authorized users ("managers", which can be set by the owner, but only to the address of the Digits token):

- `transferDai`
- `swapTokensForDai`
- `addLiquidity`
- `distributeDividends`
- `setLiquidityWallet`
- `updateUniswapV2Router`

2.2.2 Issues & Recommendations

Issue #19	**Gov Privilege: Owner can add any address as an authorized manager to withdraw funds or turn the Digits token into a honeypot**
Severity	HIGH SEVERITY
Description	The owner can call `addManager` to authorize any address as a manager, including a user wallet or smart contract, which would then have the ability to perform a number of malicious actions that should only be performed by the Digits token, including: withdrawing DAI, updating the liquidity receiving wallet and updating the router.
Recommendation	Consider removing the managers mapping and only allowing the Digits token to be able to call these functions.
Resolution	RESOLVED Only the `tokenAddress` (the Digits token) can be added as a manager.

Issue #20	**Gov Privilege: Owner can update token address, turning the Digits token into a honeypot**
Severity	MEDIUM SEVERITY
Description	The owner can call `setTokenAddress` to update the `tokenAddress` used in swapping and liquidity adding, including to the zero address or a malicious contract that would revert calls, preventing sales.
Recommendation	Consider removing the `setTokenAddress` function. The token address is already set in the constructor, so if a new token is deployed, then a new TokenStorage contract can be deployed with it.
Resolution	RESOLVED This function has been removed.

Issue #21	**LP tokens from `addLiquidity` are sent to `liquidityWallet`**
Severity	MEDIUM SEVERITY

Location	Line ~69
	`uniswapV2Router.addLiquidity(`
	`address(tokenAddress),`
	`dai,`
	`tokens,`
	`dais,`
	`0, // slippage unavoidable`
	`0, // slippage unavoidable`
	`liquidityWallet,`
	`block.timestamp`
	`);`

Description	The LP tokens provided from the `addLiquidity` call are sent to the `liquidityWallet`, who will be able to decompose the LPs and obtain the underlying tokens.
Recommendation	Consider setting the `to` address for the liquidity call to a burn address, ensuring that the received liquidity pool tokens are locked forever.
Resolution	ACKNOWLEDGED The team plans to lock the initial liquidity from this wallet with RugDoc, and will retain this generated liquidity in case it is needed later in the project's lifecycle.

Issue #22	**`setLiquidityWallet` and `updateUniswapV2Router` lack zero address checks**
Severity	MEDIUM SEVERITY
Description	The above functions lack zero address checks on `_liquidityWallet` and `newAddress` respectively. Setting these to the zero address could cause Digits swaps to fail or turn the token into a honeypot.
Recommendation	Consider ensuring these addresses can not be set to the zero address.
Resolution	RESOLVED

Issue #23	`success` is unused in `distributeDividends`
Severity	INFORMATIONAL
Description	Variables defined in a contract but not used within said contract could confuse third-party auditors. They also increase the contract length and bytecode size unnecessarily.
Recommendation	Consider removing the variable to keep the contract short and simple.
Resolution	RESOLVED The variable has been removed.

Issue #24	`dividendTracker` should be made immutable
Severity	INFORMATIONAL
Description	Variables that are defined within the constructor but further remain unchanged should be marked as immutable to save gas and to ease the reviewing process of third-parties.
Recommendation	Consider marking this variable as immutable.
Resolution	RESOLVED

Issue #25	`dai` can be made `constant`
Severity	INFORMATIONAL
Description	Variables that are never modified can be indicated as such with the `constant` keyword. This is considered best practice since it makes the code more accessible for third-party reviewers and saves gas.
Recommendation	Consider making the variable explicitly constant.
Resolution	RESOLVED

Issue #26	A number of variables can be made `external`
Severity	INFORMATIONAL
Description	Functions that are not used within the contract but only called externally can be indicated as such with the `external` keyword. This can in certain cases reduce gas and can ease the reviewing process of third parties: - `transferDai` - `swapTokensForDai` - `addLiquidity` - `distributeDividends` - `setLiquidityWallet` - `updateUniswapV2Router`
Recommendation	Consider marking the above functions as external.
Resolution	RESOLVED

2.3 DividendTracker

DividendTracker receives DAI distributed to it from the TokenStorage contract and maintains the balances of token holding users. It is a non-transferrable ERC20 token, minted and burned depending on the Digits balance of the user. It is also responsible for the swapping logic to facilitate dividend compounding.

2.3.1 Privileged Roles

The following functions can be called only by the constructor of the contract (which is the Digits token):

- `renounceOwnership`
- `transferOwnership`
- `setBalance`
- `excludeFromDividends`
- `processAccount`
- `compoundAccount`

2.3.2 Issues & Recommendations

Issue #27	Ambiguous error in `distributeDividends`
Severity	INFORMATIONAL
Location	Line 52 `require(_totalSupply > 0);`
Description	This statement does not revert with an error message, instead it reverts ambiguously, leaving users to potentially wonder what happened with their transaction. This also makes writing coverage tests difficult as the reversion method cannot be explicitly checked.
Recommendation	Consider adding an explicit reversion message to the above location.
Resolution	RESOLVED

Issue #28	`tokenAddress` should be made immutable
Severity	INFORMATIONAL
Description	Variables that are defined within the constructor but further remain unchanged should be marked as immutable to save gas and to ease the reviewing process of third-parties.
Recommendation	Consider marking this variable as `immutable`.
Resolution	RESOLVED

Issue #29	`dai`, `_name` and `_symbol` can be made constant
Severity	INFORMATIONAL
Description	Variables that are never modified can be indicated as such with the constant keyword. This is considered best practice since it makes the code more accessible for third-party reviewers and saves gas.
Recommendation	Consider making the above variables as explicitly constant.
Resolution	RESOLVED

Issue #30	A number of variables can be made external
Severity	INFORMATIONAL
Description	Functions that are not used within the contract but only called externally can be indicated as such with the external keyword. This can in certain cases reduce gas and can ease the reviewing process of third parties: - `distributeDividends` - `isExcludedFromDividends` - `processAccount` - `compoundAccount` - `withdrawnDividendOf` - `getAccountInfo` - `getLastClaimTime` - `name` - `symbol` - `decimals` - `updateUniswapV2Router`
Recommendation	Consider marking the above functions as external.
Resolution	RESOLVED


PALADIN
BLOCKCHAIN SECURITY