DESSERT FINANCE

Accumulator Finance

ERC-20 Audit

Performed at block **366446**

PERFORMED BY DESSERT FINANCE
FOR STRATEGYMAGICSEA CONTRACT CONTRACT ADDRESS: 0x735e578C211D3Ebf6AA8eAED5E2B7A9645ea53Fe
FOR VAULTV7 CONTRACT ADDRESS: 0x0661e50AAcaa235eDe7C1A167f4e679cE860Fe11

VERIFY THIS REPORT IN THE @DESSERTSWAP TELEGRAM, CLICK HERE

INITIAL DISCLAIMER

Dessert Finance provides due-diligence project audits for various projects. Dessert Finance in no way guarantees that a project will not remove liquidity, sell off team supply, or otherwise exit scam.

Dessert Finance does the legwork and provides public information about the project in an easy-to-understand format for the common person.

Agreeing to an audit in no way guarantees that a team will not remove ***all*** liquidity (“Rug Pull”), remove liquidity slowly, sell off tokens, quit the project, or completely exit scam. There is also no way to prevent private sale holders from selling off their tokens. It is ultimately your responsibility to read through all documentation, social media posts, and contract code of each individual project to draw your own conclusions and set your own risk tolerance.

Dessert Finance in no way takes responsibility for any losses, nor does Dessert Finance encourage any speculative investments. The information provided in this audit is for information purposes only and should not be considered investment advice. Dessert Finance does not endorse, recommend, support, or suggest any projects that have been audited. An audit is an informational report based on our findings, We recommend you do your own research, we will never endorse any project to invest in.

Table of Contents

Contract Code Audit – Token Overview
StrategyMagicSea

Contract Name

STRATEGYMAGICSEA

Contract Address

0X735E578C211D3EBF6AA8EAED5E2B7A9645EA53FE

Compiler Version

^0.8.0;

Optimization Enabled

TRUE

Contract Deployer Address

0X21C578EBD0C047DC3477C395ECE98D56981D4D87

EVM Version

Paris

ERC-20 Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on StrategyMagicSea

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

import "@openzeppelin-4/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin-4/contracts/token/ERC20/utils/SafeERC2

import "../../interfaces/common/IUniswapRouterETH.sol";
import "../../interfaces/common/IUniswapV2Pair.sol";
import "../../interfaces/common/IWrappedNative.sol";
import "../../interfaces/magicsea/IMasterChef.sol";
import "../Common/StratFeeManagerInitializable.sol";

contract StrategyMagicSea is StratFeeManagerInitializable {
    using SafeERC20 for IERC20;

    // Tokens used
    address public native;
    address public output;
    address public want;
    address public lpToken0;
    address public lpToken1;

    // Third party contracts
    address public chef;
    uint256 public poolId;

    bool public harvestOnDeposit;
    uint256 public lastHarvest;
    string public pendingRewardsFunctionName;

    // Routes
    address[] public outputToNativeRoute;
    address[] public outputToLp0Route;
    address[] public outputToLp1Route;
```

Contract Address
0x735e578C211D3Ebf6AA8eAED5E2B7A9645ea53Fe

Verified At
Jul 18 2024 16:41:58 PM (-04:00 UTC)

Contract Creator
0x21C578EBD0c047DC3477c395eCE98D56981D4d87

Source Code File Path
contracts/BIFI/strategies/MagicSea/StreategyMagicSea.sol

Contract Name
StrategyMagicSea

Other Settings
N/A

Compiler Version
v0.8.23+commit.f704f362

Optimization Enabled
true w/200 runs

Code is truncated to fit the constraints of this document.
The code in its entirety can be viewed here.

The contract code is **verified** on evm.iota.org.

Solidity Contract Code Audit – StrategyMagicSea (1/2)

Item	Description	Risk Level
Solidity assert violation	The contract does not use assert, so no risk of assert violations.	Low
Integer overflow in arithmetic operation	Solidity 0.8.x and above have built-in overflow checks. No unchecked arithmetic operations.	Low
Integer underflow in arithmetic operation	Solidity 0.8.x and above have built-in underflow checks. No unchecked arithmetic operations.	Low
Caller can redirect execution to arbitrary locations	No use of delegatecall.	Low
Caller can write to arbitrary storage locations	Proper use of onlyOwner and other access controls mitigate this risk.	Low
Dangerous use of uninitialized storage variables	No use of uninitialized storage variables.	Low
Any sender can withdraw mainnet currency from the contract account	No functionality allows withdrawal of native currency by arbitrary users.	Low
Any sender can trigger Authorization Control	Proper use of onlyOwner and whenNotPaused modifiers.	Low
Use of "tx.origin" as a part of authorization control	tx.origin is used in _harvest but not for access control, which is a safe context.	Low
Delegatecall to a user-supplied address	No use of delegatecall.	Low
Call to a user-supplied address	External calls are to known addresses or checked within the context.	Low
Block timestamp influences a control flow decision	Use of block.timestamp for time checks is minimal and acceptable.	Low
Loop over unbounded data structure	Loops over rewards array are bounded by the array size which is controlled.	Low

The contract code is verified on evm.iota.org

The vulnerabilities listed above were not found in the token’s Smart Contract.

Solidity Contract Code Audit – StrategyMagicSea (2/2)

Item	Description	Risk Level
Usage of "continue" in "do-while"	No use of continue in do-while loops.	Low
Persistent state read following external call	State reads are properly managed and not following external calls.	Low
Persistent state write following external call	State writes are properly managed and not following external calls.	Low
Account state accessed after call to user-defined address	Proper checks and balances in place to ensure safety.	Low
Return value of an external call is not checked	External calls properly handle return values.	Low
Potential weak source of randomness	No use of randomness in the contract.	Low
Requirement violation	Proper use of require statements to enforce rules.	Low
Call with hardcoded gas amount	No use of hardcoded gas amounts in call functions.	Low
Incorrect token implementation	No evidence of incorrect token implementation.	Low
Function parameter shadows a state variable	Function parameters do not shadow state variables.	Low
Unary operation directly after assignment	No use of unary operations directly after assignment.	Low
Unary operation without effect	No use of unary operations without effect.	Low
Unused state variable	No unused state variables detected.	Low
Unused local variable	No unused local variables detected.	Low
Function visibility is not set	All functions have explicitly set visibility.	Low
Use of deprecated functions: callcode(), sha3(), etc...	No use of deprecated functions.	Low
Use of deprecated global variables (msg.gas, ...)	No use of deprecated global variables.	Low
Use of deprecated keywords (throw, var)	No use of deprecated keywords.	Low
Incorrect function state mutability	All functions have correct state mutability annotations.	Low

The contract code is verified on evm.iota.org

The vulnerabilities listed above were not found in the token’s Smart Contract.

Contract Code Audit – Contract Ownership -StrategyMagicSea

Contract Ownership has not been renounced at the time of Audit

The contract ownership is not currently renounced.

We have placed the contract owner address below for your viewing:

0x21C578EBD0c047DC3477c395eCE98D56981D4d87

The address above has authority over the ownable functions within the contract.

This allows the owner to call certain functions within the contract. Any compromise to the owner wallet may allow these privileges to be exploited.

Contract Code Audit – Manager Accessible Functions – StrategyMagicSea

Function Name	Parameters	Visibility
managerHarvest		external
setHarvestOnDeposit	bool _harvestOnDeposit	external
panic		public
pause		public
unpause		external
setReward	address[] calldata _rewardRoute	external
resetRewards		external

The functions listed above can be called by the contract manager.

If contract ownership has been renounced there is no way for the above listed functions to be called.

Contract Code Audit – Token Overview
VAULTV7

Contract Name

ACCUVAULTV7

Contract Address

0X0661E50AACAA235EDE7C1A167F4E679CE860FE11

Compiler Version

V0.8.23+COMMIT.F704F362

Optimization Enabled

TRUE

Contract Deployer Address

0X21C578EBD0C047DC3477C395ECE98D56981D4D87

EVM Version

Paris

ERC-20 Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on AccuVaultV7

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

import "@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradea
import "@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC
import "@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable
import "@openzeppelin/contracts-upgradeable/security/ReentrancyGuardU

import "../interfaces/beefy/IStrategyV7.sol";

/**
 * @dev Implementation of a vault to deposit funds for yield optimizi
 * This is the contract that receives funds and that users interface
 * The yield optimizing strategy itself is implemented in a separate
 */
contract AccuVaultV7 is ERC20Upgradeable, OwnableUpgradeable, Reentra
    using SafeERC20Upgradeable for IERC20Upgradeable;

    struct StratCandidate {
        address implementation;
        uint proposedTime;
    }

    // The last proposed strategy to switch to.
    StratCandidate public stratCandidate;
    // The strategy currently in use by the vault.
    IStrategyV7 public strategy;
    // The minimum time it has to pass before a strat candidate can t
    uint256 public approvalDelay;

    event NewStratCandidate(address implementation);
    event UpgradeStrat(address implementation);

    /**
```

Contract Implementation Address
0x5f05F740e5a9b8172aeAD47390E15867d46FC56E

Verified At
Jul 12 2024 17:39:38 PM (-04:00 UTC)

Contract Creator
0x21C578EBD0c047DC3477c395eCE98D56981D4d87

Source Code File Path
contracts/BIFI/vaults/AccuVaultV7.sol

Contract Name
AccuVaultV7

Other Settings
N/A

Compiler Version
v0.8.23+commit.f704f362

Optimization Enabled
true w/200 runs

Code is truncated to fit the constraints of this document.
The code in its entirety can be viewed here.

The contract code is **verified** on evm.iota.org.

Solidity Contract Code Audit – AccuVaultV7 (1/2)

Item	Description	Risk Level
Solidity assert violation	The contract does not use assert, so no risk of assert violations.	Low
Integer overflow in arithmetic operation	Solidity 0.8.x and above have built-in overflow checks. No unchecked arithmetic operations.	Low
Integer underflow in arithmetic operation	Solidity 0.8.x and above have built-in underflow checks. No unchecked arithmetic operations.	Low
Caller can redirect execution to arbitrary locations	No use of delegatecall.	Low
Caller can write to arbitrary storage locations	Proper use of onlyOwner and other access controls mitigate this risk.	Low
Dangerous use of uninitialized storage variables	No use of uninitialized storage variables.	Low
Any sender can withdraw native currency (ETH/IOTA) from the contract account	No functionality allows withdrawal of native currency by arbitrary users.	Low
Any sender can trigger Authorization Control	Proper use of onlyOwner and whenNotPaused modifiers.	Low
Use of "tx.origin" as a part of authorization control	No use of tx.origin for authorization control.	Low
Delegatecall to a user-supplied address	No use of delegatecall.	Low
Call to a user-supplied address	External calls are to known addresses or checked within the context.	Low
Block timestamp influences a control flow decision	Use of block.timestamp for time checks is minimal and acceptable.	Low
Loop over unbounded data structure	No unbounded loops detected.	Low

The contract code is verified on evm.iota.org.

The vulnerabilities listed above were not found in the token’s Smart Contract.

Solidity Contract Code Audit – AccuVaultV7 (2/2)

Item	Description	Risk Level
Usage of "continue" in "do-while"	No use of continue in do-while loops.	Low
Persistent state read following external call	State reads are properly managed and not following external calls.	Low
Persistent state write following external call	State writes are properly managed and not following external calls.	Low
Account state accessed after call to user-defined address	Proper checks and balances in place to ensure safety.	Low
Return value of an external call is not checked	External calls properly handle return values.	Low
Potential weak source of randomness	No use of randomness in the contract.	Low
Requirement violation	Proper use of require statements to enforce rules.	Low
Call with hardcoded gas amount	No use of hardcoded gas amounts in call functions.	Low
Incorrect token implementation	No evidence of incorrect token implementation.	Low
Function parameter shadows a state variable	Function parameters do not shadow state variables.	Low
Unary operation directly after assignment	No use of unary operations directly after assignment.	Low
Unary operation without effect	No use of unary operations without effect.	Low
Unused state variable	No unused state variables detected.	Low
Unused local variable	No unused local variables detected.	Low
Function visibility is not set	All functions have explicitly set visibility.	Low
Use of deprecated functions: callcode(), sha3(), etc.	No use of deprecated functions.	Low
Use of deprecated global variables (msg.gas, ...)	No use of deprecated global variables.	Low
Use of deprecated keywords (throw, var)	No use of deprecated keywords.	Low
Incorrect function state mutability	All functions have correct state mutability annotations.	Low

The contract code is verified on evm.iota.org.

The vulnerabilities listed above were not found in the token’s Smart Contract.

Contract Code Audit – Contract Ownership - AccuVaultV7

Contract Ownership has not been renounced at the time of Audit

The contract ownership is not currently renounced.

We have placed the contract owner address below for your viewing:

0x21C578EBD0c047DC3477c395eCE98D56981D4d87

The address above has authority over the ownable functions within the contract.

This allows the owner to call certain functions within the contract. Any compromise to the owner wallet may allow these privileges to be exploited.

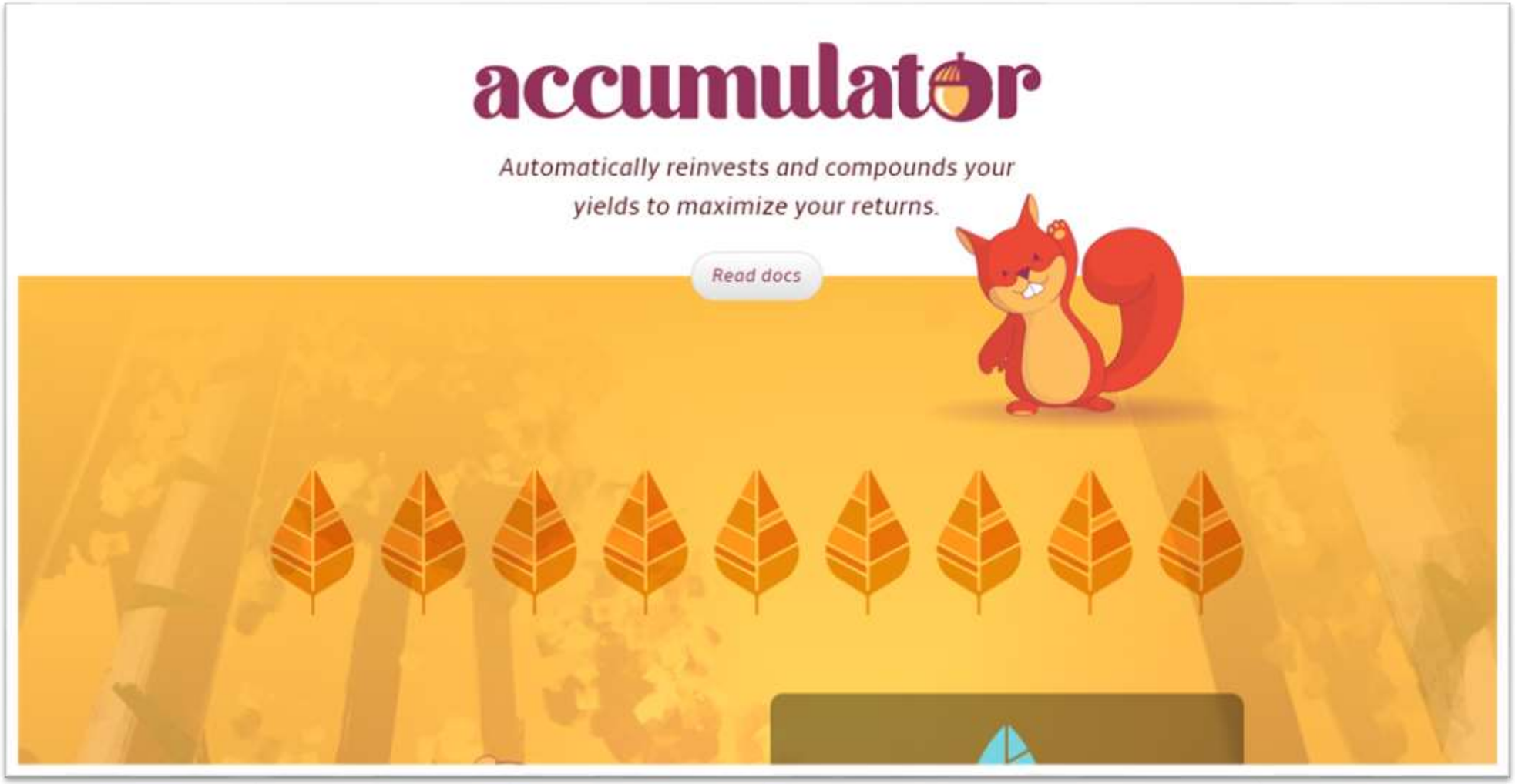
Contract Code Audit – Owner Accessible Functions – AccuvaultV7

Function Name	Parameters	Visibility	Audit Notes
proposeStrat	address _implementation	public	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
upgradeStrat		public	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
inCaseTokensGetStuck	address _token	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.

The functions listed above can be called by the contract owner.

If contract ownership has been renounced there is no way for the above listed functions to be called.

Website Part 1 – Overview
www.accumulator.finance

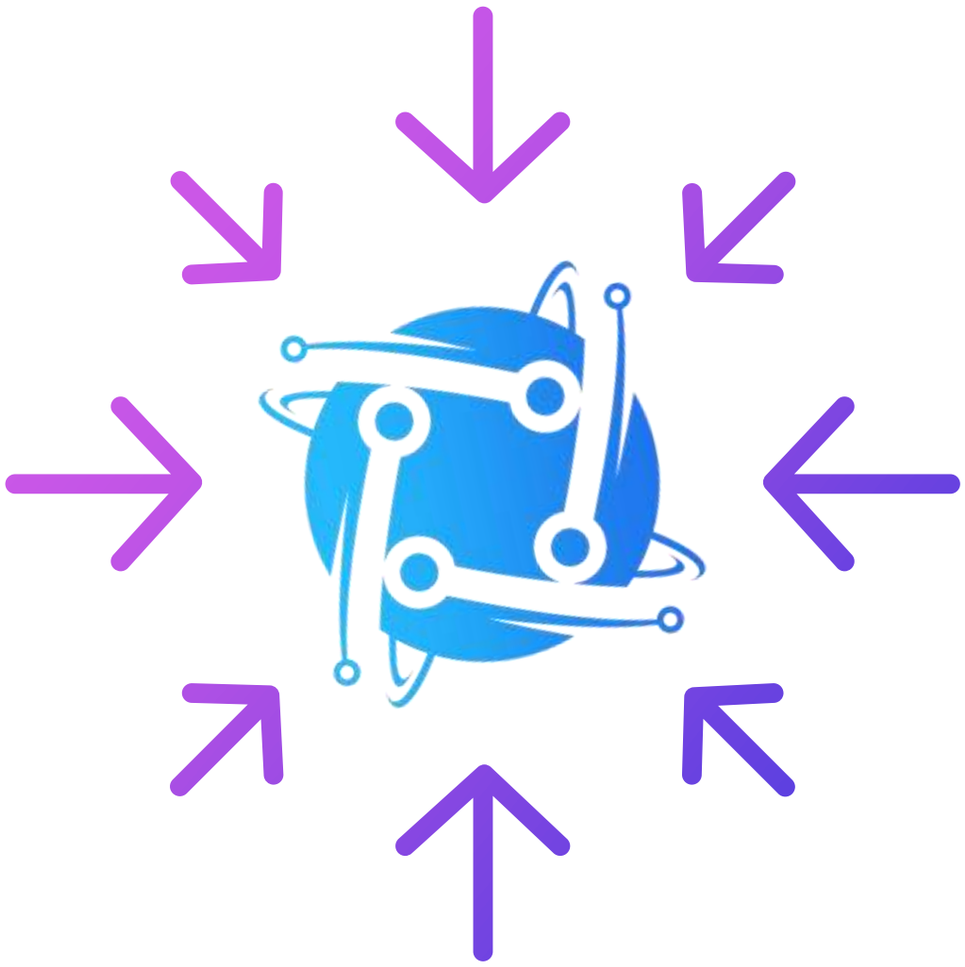


Above images are actual snapshots of the current live website of the project.

Website was registered on 04/28/2023, registration expires 04/28/2024.

✗ This meets the 3 year minimum we like to see on new projects.

Website Part 2 – Checklist



- ✓ Mobile Friendly
- ✓ No JavaScript Errors
- ✓ Spell Check
- ✓ SSL Certificate

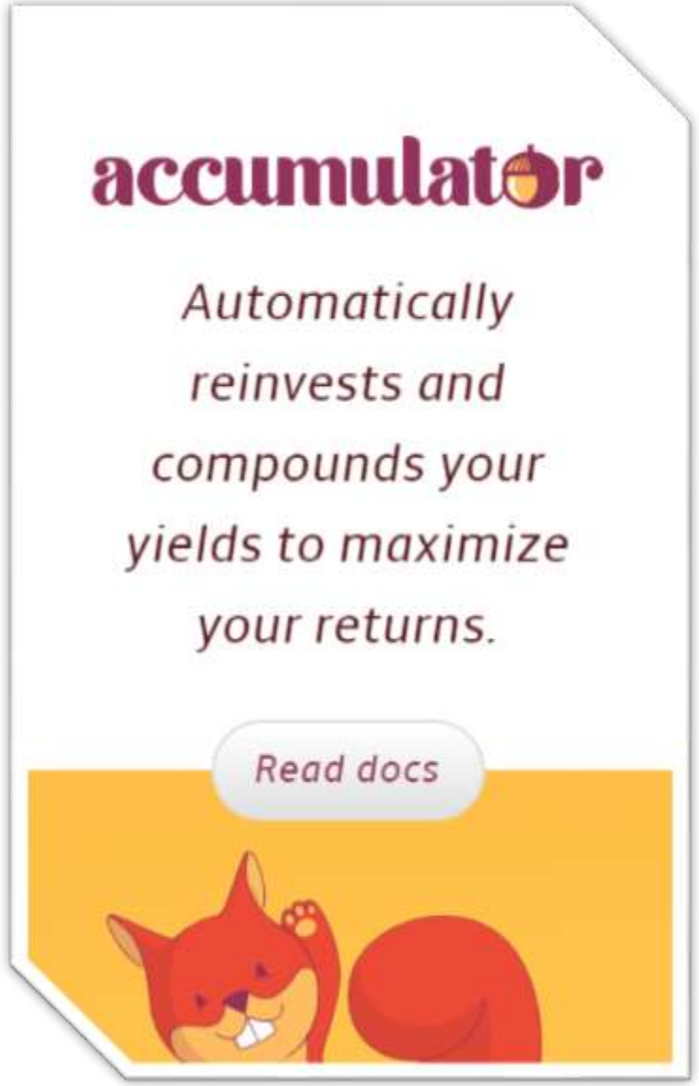
The website contained no JavaScript errors. No typos, or grammatical errors were present, and we found a valid SSL certificate allowing for access via https.

No additional issues were found on the website.

Website Part 3 – Responsive HTML5 & CSS3

No issues were found on the Mobile Friendly check for the website. All elements loaded properly and browser resize was not an issue. The team has put a considerable amount of thought and effort into making sure their website looks great on all screens.

No severe JavaScript errors were found. No issues with loading elements, code, or stylesheets.



Website Part 4 (GWS) – General Web Security

SSL CERTIFICATE

A valid SSL certificate was found. Details are as follows:

Offered to: accumulator.finance

Issued by: R3

Valid Until: Jan 2024

✓

CONTACT EMAIL

A valid contact email was found on the official website. Contact email is listed as shown below:

Contact

N/A

✗

SPAM / MALWARE / POPUPS

No malware found

No injected spam found

No internal server errors

No popups found

Domain is marked clean by Google, McAfee, Sucuri Labs, & ESET

✓

Social Media

We were able to locate a variety of Social Media networks for the project.

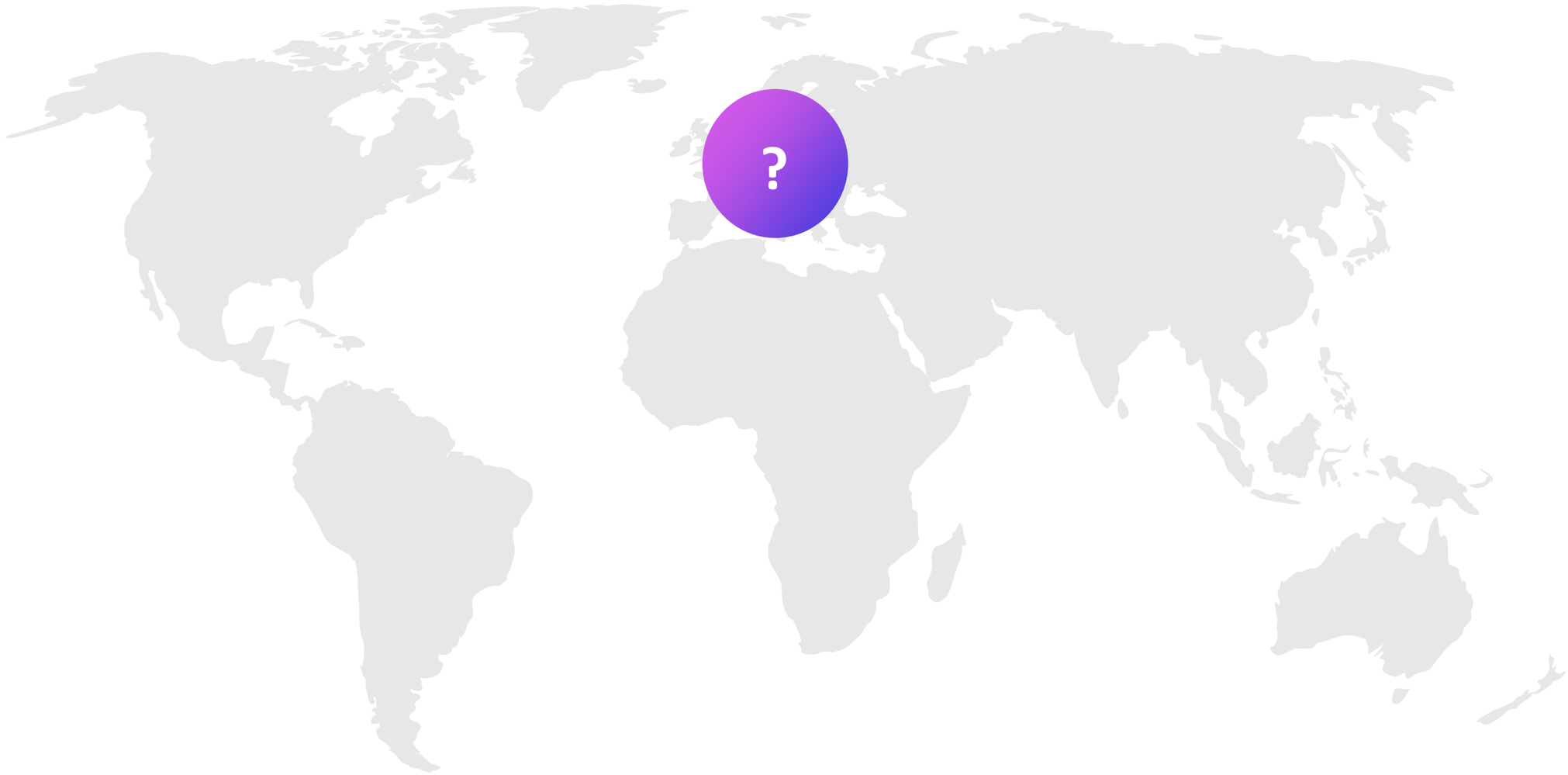
All links have been conveniently placed below.

Twitter Medium Discord

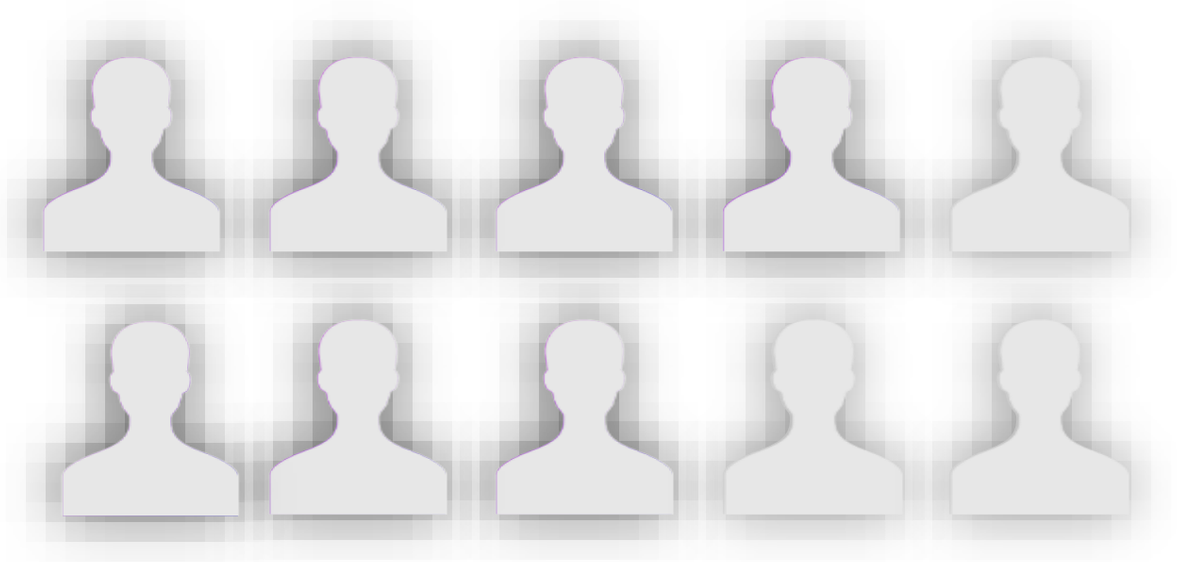
✓ **At least 3 social media networks were found.**

Location Audit

We were unable to identify a primary location for the project at this time or a location has not been declared.



Team Overview



We are unable to find any information about the team on the website at this time. Projects may choose to stay anonymous for a myriad of reasons.

Roadmap

A roadmap was found on the official website, we have conveniently placed it on this page for your viewing.

Disclaimer

The opinions expressed in this document are for general informational purposes only and are **not intended to provide specific advice or recommendations for any individual or on any specific investment**. It is only intended to provide education and public knowledge regarding projects. This audit is only applied to the type of auditing specified in this report and the scope of given in the results. Other unknown security vulnerabilities are beyond responsibility. Dessert Finance only issues this report based on the attacks or vulnerabilities that already existed or occurred before the issuance of this report. For the emergence of new attacks or vulnerabilities that exist or occur in the future, Dessert Finance lacks the capability to judge its possible impact on the security status of smart contracts, thus taking no responsibility for them. The smart contract analysis and other contents of this report are based solely on the documents and materials that the contract provider has provided to Dessert Finance or was publicly available before the issuance of this report (issuance of report recorded via block number on cover page), if the documents and materials provided by the contract provider are missing, tampered, deleted, concealed or reflected in a situation that is inconsistent with the actual situation, or if the documents and materials provided are changed after the issuance of this report, Dessert Finance assumes no responsibility for the resulting loss or adverse effects. Due to the technical limitations of any organization, this report conducted by Dessert Finance still has the possibility that the entire risk cannot be completely detected. Dessert Finance disclaims any liability for the resulting losses.

Dessert Finance provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Even projects with a low risk score have been known to pull liquidity, sell all team tokens, or exit-scam. Please exercise caution when dealing with any cryptocurrency related platforms.

The final interpretation of this statement belongs to Dessert Finance.

Dessert Finance highly advises against using cryptocurrencies as speculative investments and they should be used solely for the utility they aim to provide.

Thank You

DESSERT FINANCE PROJECT AUDIT HAS BEEN COMPLETED FOR **ACCUMULATOR FINANCE** AT BLOCK NUMBER: **366446**

THIS AUDIT IS ONLY VALID IF VIEWED ON HTTPS://WWW.DESSERTSWAP.FINANCE

www.dessertswap.finance
https://t.me/dessertswap