



**SOLIDProof**  
*Bring trust into your projects*

**Blockchain Security | Smart Contract Audits | KYC**

MADE IN GERMANY

# Audit

**Security Assessment**  
**10. January, 2022**

**For**



Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	10
Risk Level	10
Capabilities	11
Scope of Work	13
Inheritance Graph	13
Verify Claims	14
Modifiers	17
CallGraph	21
Source Units in Scope	22
Critical issues	23
High issues	23
Medium issues	23
Low issues	23
Informational issues	24
Commented Code exist	24
Audit Comments	24
SWC Attacks	25

# Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

**SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.**

SolidProof.io Reports represent an extensive auditing 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. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	10. January 2022	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>

## **Network**

Binance Smart Chain (BEP20)

## **Website**

<https://devilfinance.io/>

## **Telegram**

<https://t.me/devilfinancechat>

## **Twitter**

[https://twitter.com/DevilFinance\\_io](https://twitter.com/DevilFinance_io)

## **Github**

<https://github.com/devilfinanceio/devil-official-contracts>

## **Discord**

<https://discord.gg/wWX6A52YZ7>

## Description

In its essence Devil Finance is a **Decentralized Yield Optimizer platform** that allows its users to earn compound interest on their crypto holdings. Devil Finance runs on the Fantom blockchain and offers one of the leading market yield strategies and operations.

Our mission is to give our users (beginner & professional) the opportunity to **save, grow, and accumulate assets** for the future **in a safe, efficient, and user friendly way**.

We will be providing access to collections of non-fiat inflationary assets. We **provide rewards in return for utilizing our platform**, namely in the form of 'staking' tokens and our native token (DEVIL).

We will collect tokens from many users and stake them on a mass scale on their behalf. Through more frequent compounding, more efficient gas utilization, and other creative automations we will **save users fees and most of all maximize their returns**.

## Project Engagement

During the 8th of January 2022, **Devil Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

## Logo



## Contract Link

**v1.0**

- As file

# Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
<b>Critical</b>	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
<b>High</b>	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
<b>Medium</b>	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
<b>Low</b>	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
<b>Informational</b>	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

# **Auditing Strategy and Techniques Applied**

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

## **Methodology**

The auditing process follows a routine series of steps:

1. Code review that includes the following:
  - i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
  - ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
  - iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.
2. Testing and automated analysis that includes the following:
  - i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
  - ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

## Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

NativeFarm

```
Context
SafeMath
IERC20
ERC20
Address
SafeERC20
EnumerableSet
Ownable
ReentrancyGuard
NativeToken
IStrategy
```

TimelockController

```
./interfaces/IERC20.sol
./libraries/SafeERC20.sol
./helpers/AccessControl.sol
./helpers/ReentrancyGuard.sol
```

INativeFarm

IStrategy

Strategy

```
./interfaces/IERC20.sol
./libraries/SafeERC20.sol
./helpers/ReentrancyGuard.sol
./helpers/Pausable.sol
./helpers/Ownable.sol
./interfaces/IXswapFarm.sol
./interfaces/IXRouter01.sol
./interfaces/IXRouter02.sol
```

• IWFTM




## Tested Contract Files


This audit covered the following files listed below with a SHA-1 Hash.

*A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.*

### v1.0

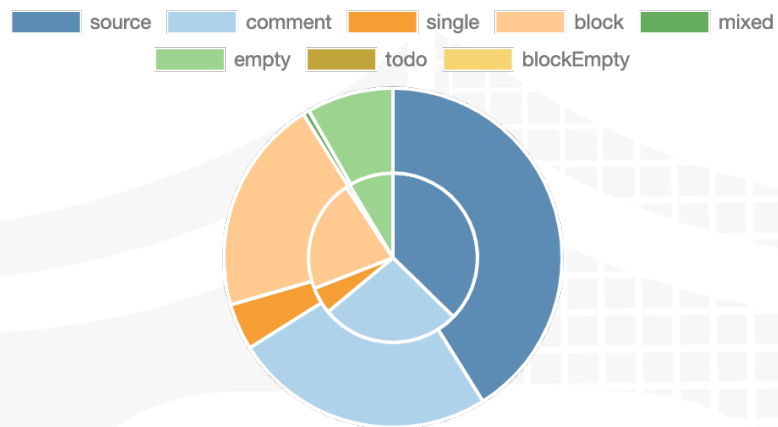


File Name	SHA-1 Hash
contracts/interfaces/IXRouter02.sol	a56a4c7451804b4658c89e6e18cd293894116e44
contracts/interfaces/IXRouter01.sol	0139cae1d2f54163fdf71df43834542a2bbe5b96
contracts/interfaces/IPancakePair.sol	828c419bbce7c9ec3a887ee533141e9565382d95
contracts/interfaces/IXswapFarm.sol	1c45ee072762a3a74788e085bfc084eb8ca438d0
contracts/interfaces/IERC20.sol	350bda155310a5f9fda08a3cc573fe08275343b2
contracts/helpers/AccessControl.sol	8d1330bb547be631266eb89ed7261b565ec383d4
contracts/helpers/Context.sol	3e98f9bcd3b23ea7bf2c4e28138c2d5335ff7398
contracts/helpers/Ownable.sol	7699c9276ddc9c270d9f454606d8262fe0818f53
contracts/helpers/Pausable.sol	2b0fde1e729d283d0a57126723e14dec98dba2bf
contracts/helpers/ReentrancyGuard.sol	d38417e818d037eb2eef1058998f5b450dd9362d
contracts/TimelockController.sol	6ff8c894c4764bc8da3230503f0437fdced5062f
contracts/NativeFarm.sol	2b50380ad5bc56f7bc8e0aefa6cd22f9b7880e90
contracts/libraries/EnumerableSet.sol	2238466f38074b887a73f03add2e43231571f592
contracts/libraries/Address.sol	d9a6eb92eabc89ea7506e63a2938ccab932db191
contracts/libraries/SafeMath.sol	c1193bc1ea44695594726881e36783e641eb5214
contracts/libraries/SafeERC20.sol	21a7ad99c1006d2f99a37cb9bbfe1692c059cb10
contracts/Strategy.sol	93c41b705a738a7ca333710bd0f3e3a1290bd1dd

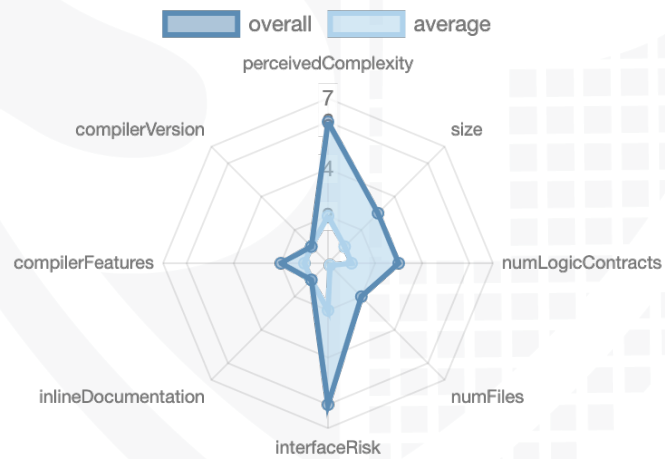


# Metrics

## Source Lines v1.0



## Risk Level v1.0



## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	4	8	10	9

### Exposed Functions

*This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.*

Version	Public	Payable
1.0	176	11

Version	External	Internal	Private	Pure	View
1.0	104	280	20	31	82

### State Variables

Version	Total	Public
1.0	76	58

### Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	0.6.12 ≥0.6.12 ^0.6.12	ABIEncoderV2	yes	yes (4 asm blocks)	

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/ Create/ Create2
1.0	yes		yes	yes		



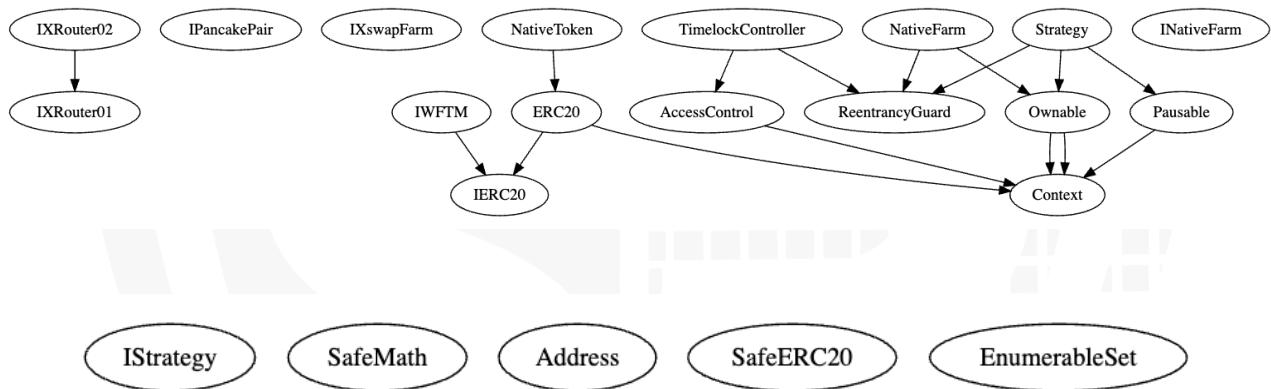
## Scope of Work

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

## Inheritance Graph v1.0



# Verify Claims

## Correct implementation of Token standard

Tested	Verified
✓	✓



## Write functions of contract

▼ STRATEGY	▼ NATIVEFARM	▼ TIMELOCKCONTROLLER
convertDustToEarned	add	add
deposit	deposit	cancel
earn	emergencyWithdraw	deleverageOnce
farm	inCaseTokensGetStuck	earn
inCaseTokensGetStuck	renounceOwnership	execute
pause	set	executeBatch
renounceOwnership	setMintParameters	executeSet
setbuyBackRate	transferOwnership	farm
setBuybackRouterAddress	updatePool	grantRole
setControllerFee	withdraw	noTimeLockFunc1
setDepositFeeFactor	withdrawAll	noTimeLockFunc2
setEntranceFeeFactor		noTimeLockFunc3
setGov		pause
setWithdrawFeeFactor		rebalance
transferOwnership		renounceRole
unpause		revokeRole
withdraw		schedule
		scheduleBatch
		scheduleSet
		setDevWalletAddress
		unpause
		updateMinDelay
		updateMinDelayReduced
		withdrawBEP20
		withdrawBNB
		wrapBNB

## Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

### Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚠
Unverified / Not checked	✗
Not available	—



# Modifiers

## NativeFarm

- ◆ add
  - Ⓜ onlyOwner
- ◆ set
  - Ⓜ onlyOwner
- ◆ updatePool
- ◆ deposit
  - Ⓜ nonReentrant
- ◆ withdraw
  - Ⓜ nonReentrant
- ◆ withdrawAll
  - Ⓜ nonReentrant
- ◆ emergencyWithdraw
  - Ⓜ nonReentrant
- ◆ inCaseTokensGetStuck
  - Ⓜ onlyOwner
- ◆ setMintParameters
  - Ⓜ onlyOwner

## TimelockController

◆ schedule

Ⓜ onlyRole

◆ scheduleBatch

Ⓜ onlyRole

◆ cancel

Ⓜ onlyRole

◆ execute 💰

Ⓜ onlyRole

Ⓜ nonReentrant

◆ executeBatch 💰

Ⓜ onlyRole

Ⓜ nonReentrant

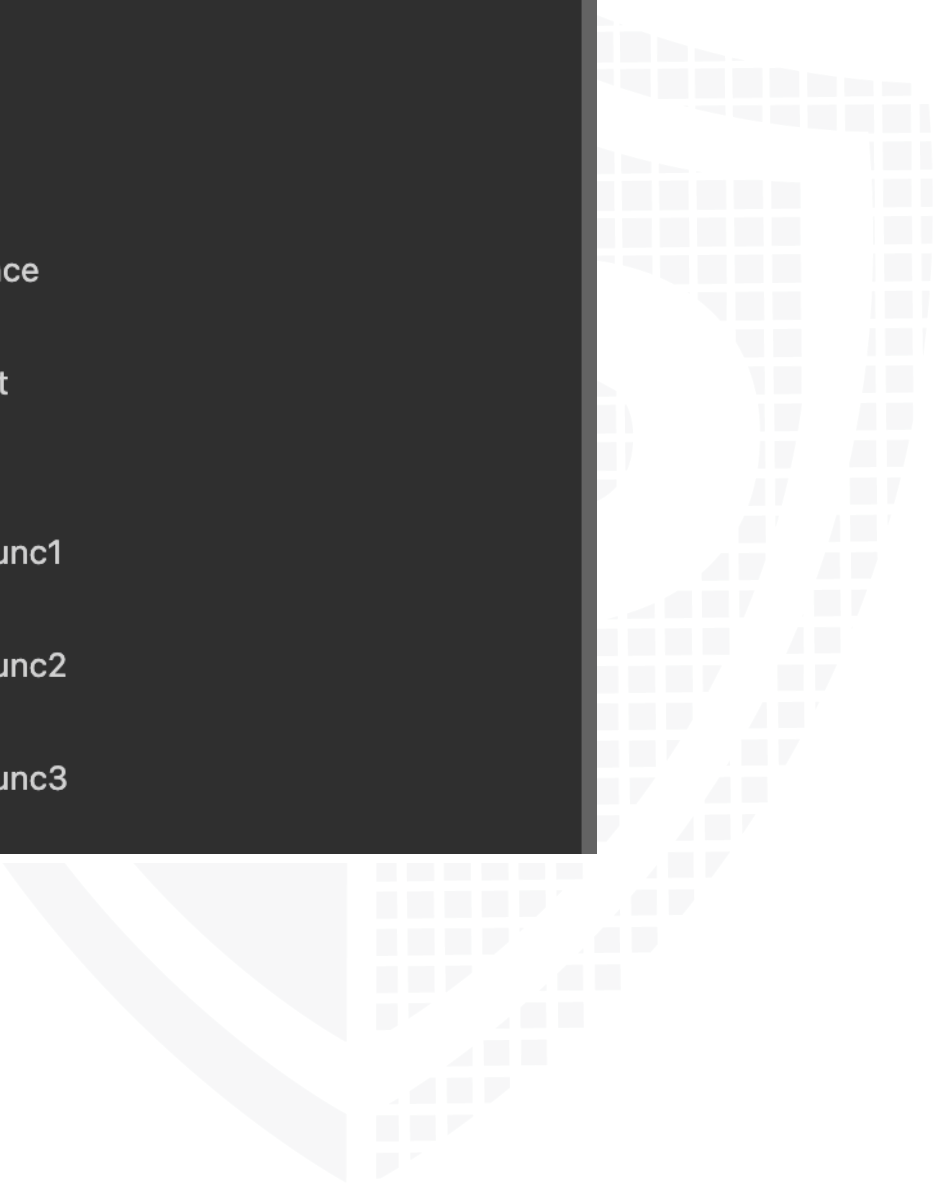
◆ scheduleSet

Ⓜ onlyRole

◆ executeSet 💰

Ⓜ onlyRole

Ⓜ nonReentrant



- ◆ add
  - Ⓜ onlyRole
- ◆ earn
  - Ⓜ onlyRole
- ◆ farm
  - Ⓜ onlyRole
- ◆ pause
  - Ⓜ onlyRole
- ◆ unpause
  - Ⓜ onlyRole
- ◆ rebalance
  - Ⓜ onlyRole
- ◆ deleverageOnce
  - Ⓜ onlyRole
  - Ⓜ nonReentrant
- ◆ wrapBNB
  - Ⓜ onlyRole
- ◆ noTimeLockFunc1
  - Ⓜ onlyRole
- ◆ noTimeLockFunc2
  - Ⓜ onlyRole
- ◆ noTimeLockFunc3
  - Ⓜ onlyRole

## Strategy

### ◆ deposit

- Ⓜ onlyOwner
- Ⓜ nonReentrant
- Ⓜ whenNotPaused

### ◆ farm

- Ⓜ nonReentrant

### ◆ withdraw

- Ⓜ onlyOwner
- Ⓜ nonReentrant

### ◆ earn

- Ⓜ nonReentrant
- Ⓜ whenNotPaused

### ◆ convertDustToEarned

- Ⓜ whenNotPaused

### ◆ pause

- Ⓜ onlyAllowGov

### ◆ unpause

- Ⓜ onlyAllowGov

### ◆ setEntranceFeeFactor

- Ⓜ onlyAllowGov

### ◆ setControllerFee

- Ⓜ onlyAllowGov

### ◆ setGov

- Ⓜ onlyAllowGov

### ◆ setDepositFeeFactor

- Ⓜ onlyAllowGov

### ◆ setWithdrawFeeFactor

- Ⓜ onlyAllowGov

### ◆ setbuyBackRate

- Ⓜ onlyAllowGov

### ◆ setBuybackRouterAddress

- Ⓜ onlyAllowGov

### ◆ inCaseTokensGetStuck

- Ⓜ onlyAllowGov

# CallGraph



# Source Units in Scope

## v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/interfaces/IXRouter02.sol	_____	1	50	6	4	_____	16	
	contracts/interfaces/IXRouter01.sol	_____	1	160	4	3	_____	48	
	contracts/interfaces/IPancakePair.sol	_____	1	52	7	5	_____	55	_____
	contracts/interfaces/IXswapFarm.sol	_____	1	28	4	3	9	13	_____
	contracts/interfaces/IERC20.sol	_____	1	86	26	21	54	13	
	contracts/helpers/AccessControl.sol	1	_____	206	202	79	101	43	
	contracts/helpers/Context.sol	1	_____	13	13	10	2	1	
	contracts/helpers/Ownable.sol	1	_____	62	62	33	21	23	_____
	contracts/helpers/Pausable.sol	1	_____	80	80	29	41	14	
	contracts/helpers/ReentrancyGuard.sol	1	_____	45	45	15	22	5	_____
	contracts/TimelockController.sol	1	2	621	500	290	173	267	
	contracts/NativeFarm.sol	10	2	1710	1407	685	674	426	
	contracts/libraries/EnumerableSet.sol	1	_____	321	274	98	141	34	
	contracts/libraries/Address.sol	1	_____	262	190	96	116	49	
	contracts/libraries/SafeMath.sol	1	_____	157	145	39	93	10	
	contracts/libraries/SafeERC20.sol	1	_____	131	106	66	31	25	
	contracts/Strategy.sol	1	1	486	457	355	38	269	
	<b>Totals</b>	<b>21</b>	<b>10</b>	<b>4470</b>	<b>3528</b>	<b>1831</b>	<b>1516</b>	<b>1311</b>	

## Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
nSLOC	normalized source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

# Audit Results

## AUDIT PASSED

### Critical issues

No critical issues

### High issues

No high issues

### Medium issues

Issue	File	Type	Line	Description
#1	NativeFarm	Unchecked tokens transfer	1693	Use `SafeERC20`, or ensure that the transfer/transferFrom return value is checked

### Low issues

Issue	File	Type	Line	Description
#1	All	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	-	We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities
#2	NativeFarm	Constant cannot be set	1708	Constant variable cannot be set in a function  NATIVEPerBlock
#3	Strategy	Contract cannot be compiled	22	Remove abstract from contract if you want to compile contract

## Informational issues

Issue	File	Type	Line	Description
#1	All	SPDX License not provided	-	Provide SPDX License to source code
#2	Strategy	Remove unused parameter	88,143	Variable was not used

## Commented Code exist

There are some instances of code being commented out in the following files that should be removed:

File	Line	Comment
NativeFarm	129	// assert(a == b * c + a % b); // There is no case in which this doesn't hold
TimelockController	163	// for (uint256 i = 0; i < proposers.length; ++i) { //   _setupRole(PROPOSER_ROLE, proposers[i]); // }
TimelockController	169	// for (uint256 i = 0; i < executors.length; ++i) { //   _setupRole(EXECUTOR_ROLE, executors[i]); // }
IXSwapFarm	15	// function pendingCake(uint256 _pid, address _user) //   external //   view //   returns (uint256);
SafeMath	116	// assert(a == b * c + a % b); // There is no case in which this doesn't hold

## Recommendation

Remove the commented code, or address them properly.

## Audit Comments

### 11. January 2022:

- Deployer can pause following contracts
  - Strategy
- Naming Convention
  - Constant variables should be uppercased
- Read whole report for more information



## SWC Attacks

ID	Title	Relationships	Status
<a href="#">SW C-1 36</a>	Unencrypted Private Data On-Chain	<a href="#">CWE-767: Access to Critical Private Variable via Public Method</a>	PASSED
<a href="#">SW C-1 35</a>	Code With No Effects	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 34</a>	Message call with hardcoded gas amount	<a href="#">CWE-655: Improper Initialization</a>	PASSED
<a href="#">SW C-1 33</a>	Hash Collisions With Multiple Variable Length Arguments	<a href="#">CWE-294: Authentication Bypass by Capture-replay</a>	PASSED
<a href="#">SW C-1 32</a>	Unexpected Ether balance	<a href="#">CWE-667: Improper Locking</a>	PASSED
<a href="#">SW C-1 31</a>	Presence of unused variables	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 30</a>	Right-To-Left-Override control character (U+202E)	<a href="#">CWE-451: User Interface (UI) Misrepresentation of Critical Information</a>	PASSED
<a href="#">SW C-1 29</a>	Typographical Error	<a href="#">CWE-480: Use of Incorrect Operator</a>	PASSED
<a href="#">SW C-1 28</a>	DoS With Block Gas Limit	<a href="#">CWE-400: Uncontrolled Resource Consumption</a>	PASSED

<a href="#">SW C-1 27</a>	Arbitrary Jump with Function Type Variable	<a href="#">CWE-695: Use of Low-Level Functionality</a>	<b>PASSED</b>
<a href="#">SW C-1 25</a>	Incorrect Inheritance Order	<a href="#">CWE-696: Incorrect Behavior Order</a>	<b>PASSED</b>
<a href="#">SW C-1 24</a>	Write to Arbitrary Storage Location	<a href="#">CWE-123: Write-what-where Condition</a>	<b>PASSED</b>
<a href="#">SW C-1 23</a>	Requirement Violation	<a href="#">CWE-573: Improper Following of Specification by Caller</a>	<b>PASSED</b>
<a href="#">SW C-1 22</a>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	<b>PASSED</b>
<a href="#">SW C-1 21</a>	Missing Protection against Signature Replay Attacks	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	<b>PASSED</b>
<a href="#">SW C-1 20</a>	Weak Sources of Randomness from Chain Attributes	<a href="#">CWE-330: Use of Insufficiently Random Values</a>	<b>PASSED</b>
<a href="#">SW C-11 9</a>	Shadowing State Variables	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>
<a href="#">SW C-11 8</a>	Incorrect Constructor Name	<a href="#">CWE-665: Improper Initialization</a>	<b>PASSED</b>
<a href="#">SW C-11 7</a>	Signature Malleability	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	<b>PASSED</b>

<a href="#">SW C-11 6</a>	Timestamp Dependence	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 5</a>	Authorization through tx.origin	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 4</a>	Transaction Order Dependence	<a href="#">CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</a>	<b>PASSED</b>
<a href="#">SW C-11 3</a>	DoS with Failed Call	<a href="#">CWE-703: Improper Check or Handling of Exceptional Conditions</a>	<b>PASSED</b>
<a href="#">SW C-11 2</a>	Delegatecall to Untrusted Callee	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 1</a>	Use of Deprecated Solidity Functions	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 0</a>	Assert Violation	<a href="#">CWE-670: Always-Incorrect Control Flow Implementation</a>	<b>PASSED</b>
<a href="#">SW C-1 09</a>	Uninitialized Storage Pointer	<a href="#">CWE-824: Access of Uninitialized Pointer</a>	<b>PASSED</b>
<a href="#">SW C-1 08</a>	State Variable Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>
<a href="#">SW C-1 07</a>	Reentrancy	<a href="#">CWE-841: Improper Enforcement of Behavioral Workflow</a>	<b>PASSED</b>
<a href="#">SW C-1 06</a>	Unprotected SELFDESTRUCT Instruction	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>

<a href="#">SW</a> <a href="#">C-1</a> <a href="#">05</a>	Unprotected Ether Withdrawal	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">04</a>	Unchecked Call Return Value	<a href="#">CWE-252: Unchecked Return Value</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">03</a>	Floating Pragma	<a href="#">CWE-664: Improper Control of a Resource Through its Lifetime</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">02</a>	Outdated Compiler Version	<a href="#">CWE-937: Using Components with Known Vulnerabilities</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">01</a>	Integer Overflow and Underflow	<a href="#">CWE-682: Incorrect Calculation</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">00</a>	Function Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>

The logo features the word "SolidProofed" in a white, handwritten-style script. The "P" is particularly large and stylized, with a long horizontal stroke that extends to the left. The background is a solid blue color with a faint, large shield emblem. The shield has a grid-like pattern on its right side and a solid blue area on its left side.

SolidProofed

**Blockchain Security | Smart Contract Audits | KYC**

A small horizontal bar representing the German flag, with black, red, and gold stripes.

MADE IN GERMANY