



# Smart Contract Audit

FOR  
**Ange**  
DATED : 12 July 23'



# AUDIT SUMMARY

**Project name - Ange**

**Date:** 12 July, 2023

**Scope of Audit-** Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

**Audit Status:** Passed

## Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	0	0	1
Acknowledged	0	0	0	0	0
Resolved	0	3	2	0	0



# USED TOOLS

---

## Tools:

### 1- Manual Review:

A line by line code review has been performed by audit ace team.

**2- BSC Test Network:** All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

### 3- Slither :

The code has undergone static analysis using Slither.

### Testnet version:

The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:

<https://testnet.bscscan.com/token/0x7115b6224d4F6Df0469b2d4737EdC082cf22118d>

---



# Token Information

---

**Token Name :** Ange

**Token Symbol:** aai

**Decimals:** 18

**Token Supply:** 100,000,000

**Token Address:** ---

**Checksum:**

4aa2711f21a9ddaabb65ee9df3bc8647b16f482

**Owner:**

---

**(at time of writing the audit)**

**Deployer:**

---

---



# TOKEN OVERVIEW

---

**Fees:**

Buy Fees: 0-10%

Sell Fees: 0-10%

Transfer Fees: 0%

---

**Fees Privilege:** No Fees

---

**Ownership:** owned

---

**Minting:** none

---

**Max Tx Amount/ Max Wallet Amount:** No

---

**Blacklist:** No

---

**Other Privileges:-** Initial distribution of the tokens

- Modifying fees
  - enabling trades
-



# AUDIT METHODOLOGY

---

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code is exercised when we run the test cases.
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

# VULNERABILITY CHECKLIST



Return values of low-level calls



**Gasless Send**



Private modifier



Using block.timestamp



Multiple Sends



Re-entrancy



Using Suicide



Tautology or contradiction



Gas Limit and Loops



Timestamp Dependence



Address hardcoded



Revert/require functions



Exception Disorder



Use of tx.origin



Using inline assembly



Integer overflow/underflow



Divide before multiply



Dangerous strict equalities



Missing Zero Address Validation



Using SHA3



Compiler version not fixed



Using throw



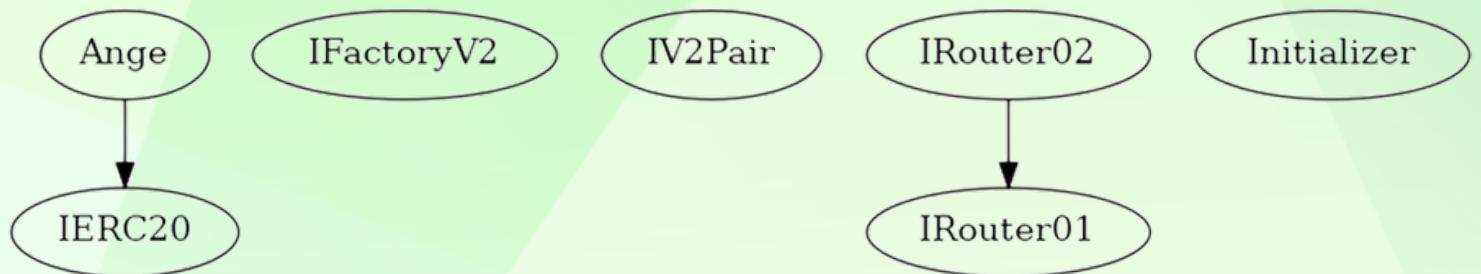
# CLASSIFICATION OF RISK

Severity	Description
◆ Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
◆ High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
◆ Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
◆ Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
◆ Gas Optimization / Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

## Findings

Severity	Found
◆ Critical	0
◆ High-Risk	3
◆ Medium-Risk	2
◆ Low-Risk	0
◆ Gas Optimization / Suggestions	1

# INHERITANCE TREE





## POINTS TO NOTE

---

- Owner is able to change buy/sell/transfer fees within 0-10%
- Owner is not able to blacklist an address
- Owner is not able to disable buy/sell/transfers
- Owner is not able to set max wallet limit and minimum wallet limits
- Owner is not able to mint new tokens
- **Owner must enable trades manually**



# CONTRACT ASSESSMENT

Contract	Type	Bases			
L   **Function Name**	**Visibility**	**Mutability**	**Modifiers**		
**IERC20**   Interface					
L   totalSupply   External	!	NO	!		
L   decimals   External	!	NO	!		
L   symbol   External	!	NO	!		
L   name   External	!	NO	!		
L   getOwner   External	!	NO	!		
L   balanceOf   External	!	NO	!		
L   transfer   External	!	●	NO	!	
L   allowance   External	!	NO	!		
L   approve   External	!	●	NO	!	
L   transferFrom   External	!	●	NO	!	
**IFactoryV2**   Interface					
L   getPair   External	!	NO	!		
L   createPair   External	!	●	NO	!	
**IV2Pair**   Interface					
L   factory   External	!	NO	!		
L   getReserves   External	!	NO	!		
L   sync   External	!	●	NO	!	
**IRouter01**   Interface					
L   factory   External	!	NO	!		
L   WETH   External	!	NO	!		
L   addLiquidityETH   External	!	\$	!	NO	!
L   addLiquidity   External	!	●	NO	!	
L   swapExactETHForTokens   External	!	\$	!	NO	!
L   getAmountsOut   External	!	NO	!		
L   getAmountsIn   External	!	NO	!		
**IRouter02**   Interface   IRouter01					
L   swapExactTokensForETHSupportingFeeOnTransferTokens   External	!	●	NO	!	
L   swapExactETHForTokensSupportingFeeOnTransferTokens   External	!	\$	!	NO	!
L   swapExactTokensForTokensSupportingFeeOnTransferTokens   External	!	●	NO	!	
L   swapExactTokensForTokens   External	!	●	NO	!	
**Initializer**   Interface					
L   setLaunch   External	!	●	NO	!	
L   getConfig   External	!	●	NO	!	
L   getInits   External	!	●	NO	!	



# CONTRACT ASSESSMENT

L   setLpPair   External !	●   NO !
L   checkUser   External !	●   NO !
L   setProtections   External !	●   NO !
L   removeSniper   External !	●   NO !
**Ange**   Implementation   IERC20	
L   <Constructor>   Public !	\$   NO !
L   transferOwner   External !	●   onlyOwner
L   renounceOwnership   External !	●   onlyOwner
L   setOperator   Public !	●   NO !
L   renounceOriginalDeployer   External !	●   NO !
L   <Receive Ether>   External !	\$   NO !
L   totalSupply   External !	NO !
L   decimals   External !	NO !
L   symbol   External !	NO !
L   name   External !	NO !
L   getOwner   External !	NO !
L   allowance   External !	NO !
L   balanceOf   Public !	NO !
L   transfer   Public !	●   NO !
L   approve   External !	●   NO !
L   _approve   Internal 🔒	●
L   approveContractContingency   External !	●   onlyOwner
L   transferFrom   External !	●   NO !
L   setNewRouter   External !	●   onlyOwner
L   setLpPair   External !	●   onlyOwner
L   setInitializer   Public !	●   onlyOwner
L   isExcludedFromFees   External !	NO !
L   setExcludedFromFees   Public !	●   onlyOwner
L   isExcludedFromProtection   External !	NO !
L   setExcludedFromProtection   External !	●   onlyOwner
L   getCirculatingSupply   Public !	NO !
L   removeSniper   External !	●   onlyOwner
L   setProtectionSettings   External !	●   onlyOwner
L   lockTaxes   External !	●   onlyOwner
L   setTaxes   External !	●   onlyOwner
L   setWallets   External !	●   onlyOwner
L   getTokenAmountAtPriceImpact   External !	NO !
L   setSwapSettings   External !	●   onlyOwner
L   setPriceImpactSwapAmount   External !	●   onlyOwner
L   setContractSwapEnabled   External !	●   onlyOwner
L   excludePresaleAddresses   External !	●   onlyOwner

# CONTRACT ASSESSMENT

	L	_hasLimits	Internal			
	L	_transfer	Internal			
	L	contractSwap	Internal			inSwapFlag
	L	_checkLiquidityAdd	Internal			
	L	enableTrading	Public			onlyOwner
	L	sweepContingency	External			onlyOwner
	L	sweepExternalTokens	External			onlyOwner
	L	multiSendTokens	External			onlyOwner
	L	finalizeTransfer	Internal			
	L	takeTaxes	Internal			

## ### Legend

	Symbol	Meaning	
----- ----- ----- -----			
		Function can modify state	
		Function is payable	



# STATIC ANALYSIS

```
Reentrancy in Ange.setNewRouter(address) (contracts/Token.sol#307-320):
  External calls:
    - lpPair = IFactoryV2(_newRouter.factory()).createPair(address(this),_newRouter.WETH()) (contracts/Token.sol#313)
  Event emitted after the call(s):
    - Approval(sender,spender,amount) (contracts/Token.sol#291)
      - _approve(address(this),address(dexRouter),type()(uint256).max) (contracts/Token.sol#319)
Reentrancy in Ange.transferOwner(address) (contracts/Token.sol#195-208):
  External calls:
    - finalizeTransfer(_owner,newOwner,balanceOf(_owner),false,false,true) (contracts/Token.sol#202)
      - initializer.checkUser(from,to,amount) (contracts/Token.sol#578-582)
  Event emitted after the call(s):
    - OwnershipTransferred(oldOwner,newOwner) (contracts/Token.sol#207)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Ange.setLpPair(address,bool) (contracts/Token.sol#322-335) uses timestamp for comparisons
  Dangerous comparisons:
    - timeSinceLastPair != 0 (contracts/Token.sol#327)
    - require(bool,string)(block.timestamp - timeSinceLastPair > 259200,3 Day cooldown.) (contracts/Token.sol#328)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Ange._checkLiquidityAdd(address,address) (contracts/Token.sol#517-531) has costly operations inside a loop:
  - hasLiqBeenAdded = true (contracts/Token.sol#524)
Ange._checkLiquidityAdd(address,address) (contracts/Token.sol#517-531) has costly operations inside a loop:
  - initializer = Initializer(address(this)) (contracts/Token.sol#526)
Ange._checkLiquidityAdd(address,address) (contracts/Token.sol#517-531) has costly operations inside a loop:
  - contractSwapEnabled = true (contracts/Token.sol#528)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Pragma version>=0.6.0<0.9.0 (contracts/Token.sol#2) is too complex
solc<0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Ange.contractSwap(uint256) (contracts/Token.sol#498-515):
  - (success,None) = marketingWallet.call(gas: 55000,value: address(this).balance)() (contracts/Token.sol#514)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IRouter01.WETH() (contracts/Token.sol#35) is not in mixedCase
Parameter Ange.setProtectionSettings(bool,bool)._antiSnipe (contracts/Token.sol#376) is not in mixedCase
Parameter Ange.setProtectionSettings(bool,bool)._antiBlock (contracts/Token.sol#376) is not in mixedCase
Constant Ange.startingSupply (contracts/Token.sol#119) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Ange._name (contracts/Token.sol#120) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Ange._symbol (contracts/Token.sol#121) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Ange._decimals (contracts/Token.sol#122) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Ange._tTotal (contracts/Token.sol#123) is not in UPPER_CASE_WITH_UNDERSCORES
Variable Ange._taxRates (contracts/Token.sol#131) is not in mixedCase
Constant Ange.maxBuyTaxes (contracts/Token.sol#133) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Ange.maxSellTaxes (contracts/Token.sol#134) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Ange.maxTransferTaxes (contracts/Token.sol#135) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Ange.masterTaxDivisor (contracts/Token.sol#136) is not in UPPER_CASE_WITH_UNDERSCORES
Variable Ange._hasLiqBeenAdded (contracts/Token.sol#153) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#47) is too similar to IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
```

**Result => A static analysis of contract's source code has been performed using slither,  
No major issues were found in the output**



# FUNCTIONAL TESTING

**Router (PCS V2):**

**0xD99D1c33F9fC3444f8101754aBC46c52416550D1**

**1- Adding liquidity (passed):**

<https://testnet.bscscan.com/tx/0xfe0b702c58767f29f621c56e3e54cf7a9910a63608e1b685627dd1e66c3748ec>

**2- Buying when excluded from fees (0% tax) (passed):**

<https://testnet.bscscan.com/tx/0x9d4922a6bed350966a93f4a9072c196b4216f2035d183ff0cb7fa251fa51883e>

**3- Selling when excluded from fees (0% tax) (passed):**

<https://testnet.bscscan.com/tx/0xf33813b0a43b1b7008571297b3878623c0d15121dd54781fe1e3be2c3f910511>

**4- Transferring when excluded from fees (0% tax) (passed):**

<https://testnet.bscscan.com/tx/0xd071802824ce90520c2d58395a64c3ec151998736088d9d3d3cfe05e7c57bcf7>

**5- Buying(0-10% tax) (passed):**

<https://testnet.bscscan.com/tx/0xb634f7b86b6de534ccbc6c0f43e002f664c8d91b092864b4b84d3eed59df0874>

**6- Selling (0-10% tax) (passed):**

<https://testnet.bscscan.com/tx/0xb634f7b86b6de534ccbc6c0f43e002f664c8d91b092864b4b84d3eed59df0874>



# FUNCTIONAL TESTING

---

**4- Transferring (0-10% tax) (passed):**

<https://testnet.bscscan.com/tx/0xab33813b6163800e9b038cbd6cda7ceedd5bc2e782a3bff4f1e1030639676639>

**4- Internal swap (ETH sent to marketing wallet / swap threshold equal to 1/1000000 of supply, the correct amount was swapped) (passed):**

<https://testnet.bscscan.com/address/0xc1610893d7770b2be156f0ffadf21efa55d7eb81#internaltx>



# FUNCTIONAL TESTING

## Centralization – Enabling Trades

Severity: **High**

**function:** enableTrading

**Status:** Resolved (contract owned by safu developer)

### Overview:

Owner of the contract must enable trades manually for investors, otherwise no one would be able to buy/sell/transfer their tokens.

```
function enableTrading() public onlyOwner {  
    require(!tradingEnabled, "Trading already enabled!");  
    require(_hasLiqBeenAdded, "Liquidity must be added.");  
    if (address(initializer) == address(0)) {  
        initializer = Initializer(address(this));  
    }  
    try initializer.setLaunch(lpPair, uint32(block.number), uint64(block.timestamp), _decimals) {} catch {}  
    try initializer.getLiqIns(balanceOf(lpPair)) returns (uint256 initThreshold, uint256 initSwapAmount) {  
        swapThreshold = initThreshold;  
        swapAmount = initSwapAmount;  
    } catch {}  
    tradingEnabled = true;  
    allowedPresaleExclusion = false;  
    launchStamp = block.timestamp;  
}
```

### Suggestion

Its suggested to either enable trades prior to presale, or transfer ownership of the contract to a certified pinsksale safu developer to guearantee enabling of trades.



# FUNCTIONAL TESTING

## Centralization – Initializer

Severity: **High**

**function:** checkUser

**Status: Resolved (contract owned by safu developer)**

### Overview:

The contract under audit uses an initializer contract to check limited wallets through the checkUser function. If the checkUser function returns false, the transaction will be reverted, giving the initializer contract the ability to essentially blacklist certain addresses.

```
if (_hasLimits(from, to) && false) {  
    bool checked;  
    try initializer.checkUser(from, to, amount) returns (bool check) {  
        checked = check;  
    } catch {  
        revert("Initializer error");  
    }  
    if (!checked) revert("You are not checked");  
}
```

However, we don't have access to the initializer contract's full implementation, especially functions like checkUser, and it's not in the scope of this audit.

### Suggestion

- The logic for checking users within the main contract could be made more transparent and auditable. This can be done by implementing the checkUser function in the main contract itself, reducing dependency on an external initializer.
- Lastly, if the initializer contract is required, provide a public and thorough audit of its code. This helps the community to understand its functionalities, providing transparency and building trust among token holders.



# FUNCTIONAL TESTING

## Centralization – Updating router

**Severity:** High

**function:** checkUser

**Status:** Resolved (contract owned by safu developer)

### Overview:

The router contract is currently upgradeable until liquidity is added. If the router is set to an invalid or malicious address, it can potentially cause the internal swaps and sell transactions leading to internal swaps to revert. Although the contract uses a try-catch block to handle the internal swap failures, it doesn't handle all types of errors, such as compiler errors (underflow/overflow).

```
function setNewRouter(address newRouter) external onlyOwner {  
    //....  
    IRouter02 _newRouter = IRouter02(newRouter);  
    //....  
}  
  
function contractSwap(uint256 contractTokenBalance) internal inSwapFlag {  
    //....  
    try dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(  
        contractTokenBalance, 0, path, address(this), block.timestamp  
    ) {} catch {  
        return;  
    }  
    //....  
}
```

### Suggestion

**Transparent and clear processes:** Make sure the process for updating the router is clear and transparent. This can be achieved by emitting events when the router is updated or providing mechanisms for stakeholders to propose and vote on updates to the router.

**validation:** validate the new router contract by implementing a function that checks the legitimacy of the contract, for instance, by ensuring it has the required interface and is not a malicious contract.



# FUNCTIONAL TESTING

## Logical – Misconfiguration of initializer variable in token contract

**Impact:** Medium

**function:** ---

**Status:** Resolved (contract owned by safu developer)

**Overview:**

In the contract under review, the initializer is set to the address of the token contract itself if it's not assigned to another address before trading is enabled. This configuration poses several significant issues in the contract that could impede its proper functioning:

1. In the finalizeTransfer function, if the initializer is set to the token address, an error will occur during the execution of the initializer.checkUser method, as it's not implemented in the contract. The contract will catch this error and revert the entire transaction.
2. Functions such as setPair, setLpPair in the contract will be uncallable due to the onlyOwner modifier. As the contract externally calls these functions, and the initializer is set to the token address, the msg.sender will be the token address, which is not equal to the contract owner.
3. Recursive calls will occur if the contract makes a call to the initializer set to its own address, which can exceed the block's gas limit, resulting in the transaction being reverted.

Although this is a medium-severity issue because the initializer can be set to a valid contract before enabling trades, it's important to note that we don't have access to the full implementation of the initializer contract, especially the checkUser function.

**Suggestion:** We recommend developing and linking an initializer contract to the main contract from the onset instead of setting it to a valid initializer after adding liquidity and enabling trades. The time between adding liquidity and enabling trades could be error-prone, e.g., mistakenly renouncing ownership, which could render the token unusable.

# FUNCTIONAL TESTING

## Logical – Excessive 45% tax if initializer is not correctly set

**Impact:** Medium

**function:** ---

**Status:** Resolved (contract owned by safu developer)

### Overview:

The current contract design leads to a 45% tax imposition on all buy/sell/transfer transactions if the initializer is not set to another address before enabling trades. This high tax rate is caused by the fact that both conditions in the `if (address(initializer) == address(this) && block.chainid != 97)` statement are true. The first condition is true when the initializer is not updated, and the second condition will always be true on any network other than the Binance Smart Chain Testnet, which has a chain ID of 97.

```
if (address(initializer) == address(this) && block.chainid != 97) currentFee = 4500;
```

The current setting of these conditions thus leads to a potential tax fee of 45%, which is considered extremely high for standard transactions.

This issue is categorized as medium severity because it can be mitigated by setting the initializer to a valid contract before enabling trades. However, note that we don't have access to the full implementation of the initializer contract.

**Suggestion:** It is recommended to develop and link an initializer contract to the main contract from the beginning rather than setting it to a valid initializer after adding liquidity and enabling trades. The period between adding liquidity and enabling trades is susceptible to errors, such as accidentally renouncing ownership, which could make the token unusable. Also, to ensure the `takeTaxes` function behaves as expected on the mainnet, the chain ID comparison value (currently set to 97) should be adjusted to match the chain ID of the intended production network. This will prevent the inadvertent application of the excessive 45% tax.



# FUNCTIONAL TESTING

## Optimizations – Redundant code

**Severity:** [Informational](#)

**function:** ---

**Status:** Not Resolved

### Overview:

Below codes are not used in other core sections of the contract.

```
address public originalDeployer;
address public operator;
```

```
// Function to set an operator to allow someone other the deployer to create things such as
launchpads.
```

```
// Only callable by original deployer.
```

```
function setOperator(address newOperator) public {
    require(msg.sender == originalDeployer, "Can only be called by original deployer.");
    address oldOperator = operator;
    if (oldOperator != address(0)) {
        _liquidityHolders[oldOperator] = false;
        setExcludedFromFees(oldOperator, false);
    }
    operator = newOperator;
    _liquidityHolders[newOperator] = true;
    setExcludedFromFees(newOperator, true);
}
```

```
function renounceOriginalDeployer() external {
```

```
    // @audit redundant
```

```
    require(msg.sender == originalDeployer, "Can only be called by original deployer.");
    setOperator(address(0));
    originalDeployer = address(0);
}
```

### Suggestion

Delete redundant code in order to reduce overall size of the contract.



# 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. 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 Auditace 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 Auditace 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 Auditace team disclaims any liability for the resulting losses.



# ABOUT AUDITACE

---

We specialize in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.



**<https://auditace.tech/>**



**[https://t.me/Audit\\_Ace](https://t.me/Audit_Ace)**



**[https://twitter.com/auditace\\_](https://twitter.com/auditace_)**



**<https://github.com/Audit-Ace>**

---