SAFUAUDIT

SMART CONTRACT SECURITY ASSESSMENT

PROJECT:

SWMETAVERSITY

DATE:

JANUARY 11, 2023

https://t.me/SafuAudit
www.safuaudit.com

Introduction

Client	SWMetaversity
Language	Solidity
Contract Address	0xbd90CCbb4B5eb18d4cCCc3525436e68468B1f280
Owner	0xEEd6F29c3c7eBDE225B7275bd53197842DbB08cA
Deployer	0xEEd6F29c3c7eBDE225B7275bd53197842DbB08cA
SHA-256 Hash	b5e69d90c748cee7fb9d2379a222e91050cd4e8f
Decimals	18
Supply	150000000
Platform	Ethereum
Compiler	v0.8.17+commit.8df45f5f
Optimization	No with 200 runs
Website	https://swdesignmetaversity.co
Twitter	
Telegram	https://t.me/swdmetaversity

Overview

Fees

- Buy fees: 0%
- Sell fees: 0%

Fees privileges

- Owner can set the buying price

Ownership

- Owned

Minting

- No

Max Tx Amount

- Can't set max Tx

Pause

- Can't pause

Blacklist

- Can't blacklist

Other Privileges

- Minting is done only when users buy Tokens, via ETH (buyViaETH function) or via USDT (buyViaUSDT function)
- The owner can modify the minting boundaries of transactions (i.e minimum and maximum amounts to be bought)

Table Of Contents

Risk Classification

Critical

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium

Issues on this level could potentially bring problems and should eventually be fixed.

Minor

Issues on this level are minor details and warning that can remain unfixed but would be better fixed at some point in the future

Informational

Information level is to offer suggestions for improvement of efficacity or security for features with a risk free factor.

Contract Inspection

File Name	SHA-1 Hash
SWMetaversity.sol	b5e69d90c748cee7fb9d2379a222e91050cd4e8f

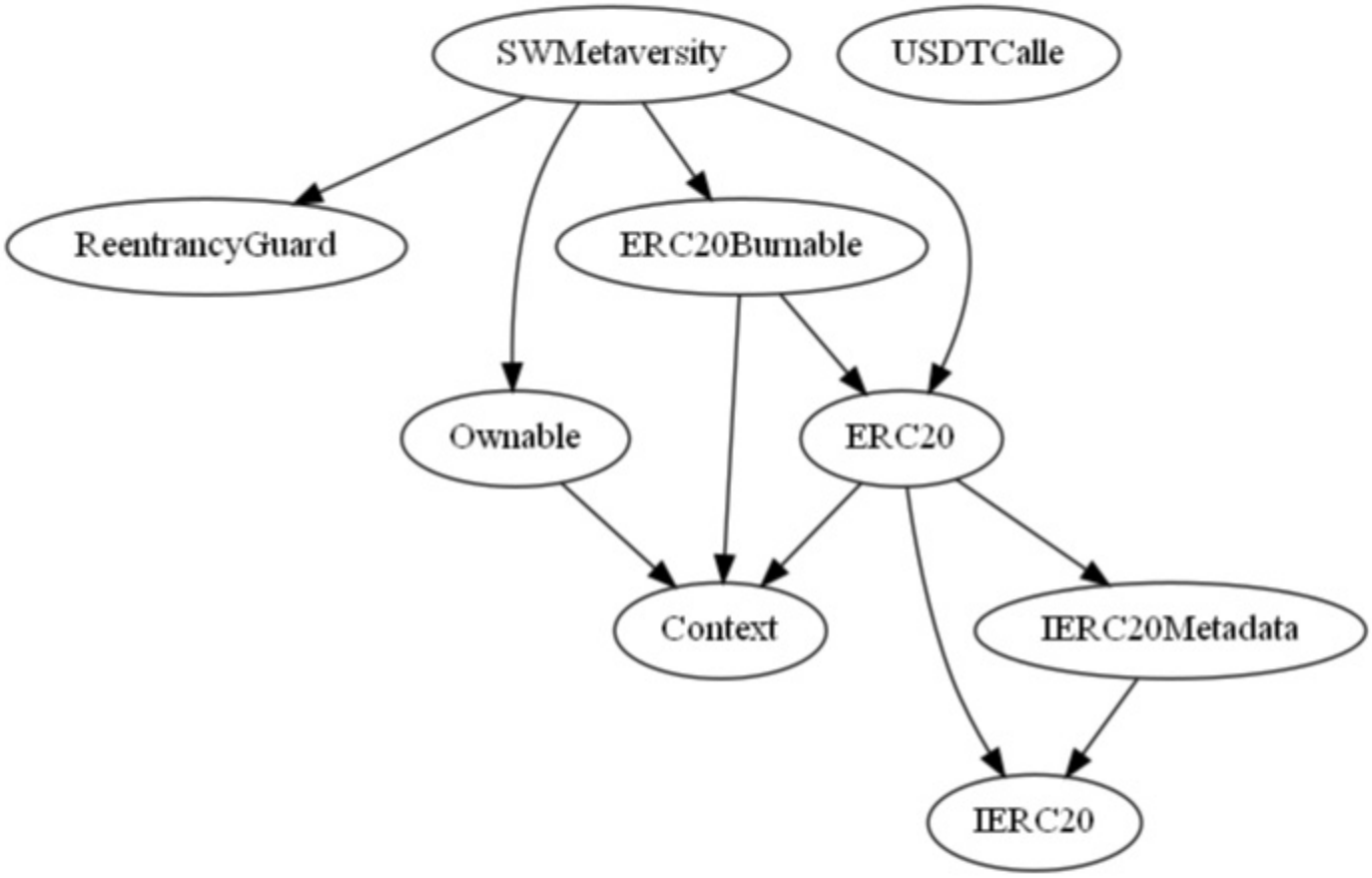
Contracts Description Table

ReentrancyGuard	Implementation			
Context	Implementation			
Ownable	Implementation	Context		
IERC20	Interface			
IERC20Metadata	Interface	IERC20		
ERC20	Implementation	Context, IERC20, IERC20Metadata		
ERC20Burnable	Implementation	Context, ERC20		
USDTCalle	Implementation			
SWMetaversity	Implementation	ERC20, ERC20Burnable, Ownable, ReentrancyGuard		
└	<Constructor>	Public ❗	🔴	ERC20
└	buyViaETH	External ❗	💵	nonReentrant
└	ethDistribution	Private 🔐	🔴	
└	buyViaUSDT	External ❗	🔴	NO ❗
└	usdtDistribution	Private 🔐	🔴	
└	updateFTETHPrice	External ❗	🔴	onlyOwner
└	updateFTUSDTPrice	External ❗	🔴	onlyOwner
└	setMaxMint	External ❗	🔴	onlyOwner
└	setMinMint	External ❗	🔴	onlyOwner
└	setMaxTokens	External ❗	🔴	onlyOwner
└	withdrawUSDTFromContract	External ❗	🔴	onlyOwner
└	withdrawERC20FromContract	External ❗	🔴	onlyOwner
└	withdraw	External ❗	🔴	onlyOwner nonReentrant

Legend

Symbol	Meaning
🔴	Function can modify state
💵	Function is payable

Contract Inheritance



Inheritance is a feature of the object-oriented programming language. It is a way of extending the functionality of a program, used to separate the code, reduces the dependency, and increases the re-usability of the existing code. Solidity supports inheritance between smart contracts, where multiple contracts can be inherited into a single contract.

Vulnerabilities Test

Test Name	Result
Function Default Visibility	Passed
Integer Overflow and Underflow	Passed
Outdated Compiler Version	Passed
Floating Pragma	Passed
Unchecked Call Return Value	Passed
Unprotected Ether Withdrawal	Passed
Unprotected SELF-DESTRUCT Instruction	Passed
Reentrancy	Passed
State Variable Default Visibility	Passed
Uninitialized Storage Pointer	Passed
Assert Violation	Passed
Use of Deprecated Solidity Functions	Passed
Delegate Call to Untrusted Callee	Passed
DoS with Failed Call	Passed
Transaction Order Dependence	Passed
Authorization through tx.origin	Passed
Block values as a proxy for time	Passed
Signature Malleability	Passed
Incorrect Constructor Name	Passed

Vulnerabilities Test

Test Name	Result
Shadowing State Variables	Passed
Weak Sources of Randomness from Chain Attributes	Passed
Missing Protection against Signature Replay Attacks	Passed
Lack of Proper Signature Verification	Passed
Requirement Violation	Passed
Write to Arbitrary Storage Location	Passed
Incorrect Inheritance Order	Passed
Insufficient Gas Griefing	Passed
Arbitrary Jump with Function Type Variable	Passed
DoS With Block Gas Limit	Passed
Typographical Error	Passed
Right-To-Left-Override control character (U+202E)	Passed
Presence of unused variables	Passed
Unexpected Ether balance	Passed
Hash Collisions With Multiple Variable Length Arguments	Passed
Message call with the hardcoded gas amount	Passed
Code With No Effects (Irrelevant/Dead Code)	Passed
Unencrypted Private Data On-Chain	Passed

Findings

ID	Category	Issue	Severity
CS-01	Coding Standards	Directly distributing the msg.value on purchase	Medium
GO-01	Gas Optimization	Unnecessary double validation	Optimization

CS-01 Directly Distributing The Msg.value On Purchase

Lines # 768

```
function buyViaETH(address to, uint256 amount) external payable nonReentrant {
   require(amount > 0, "Token Amount Should be greater than zero");
   require(totalSupply() < MAX_SWDM_TOKENS,"All SWDM Tokens Have been minted.");
   require(totalSupply() + amount * 10 ** decimals() <= MAX_SWDM_TOKENS,
"Token amount exceeds with available SWDM FT's.");
   require(amount >= minMint,"You can't mint less than the minimum mint count.");
   require(amount <= maxMint,"You have exceeded maximum mint count.");
   require(msg.value >= ftPrice * amount,
"Insuffient ETH amount sent For Purhcase SWDM FTs.");
            ethDistribution(msg.value);
            _mint(to, amount * 10 ** decimals());
        }
```

Description

The buyViaETH() function is directly distributing the msg.value of the transaction to the ethDistribution() function, keeping the excess balance of the transaction - thus not being returned to the user.

Recommendation

It is recommended in these cases to always return any excess balance to the user.

GO-01 Unnecessary Double Validation

Lines # 778, 779

```
-> require(totalSupply() < MAX_SWDM_TOKENS,"All SWDM Tokens Have been minted.");
-> require(totalSupply() + amount * 10 ** decimals() <= MAX_SWDM_TOKENS,
"Token amount exceeds with available SWDM FT's.");
```

Description

Within the buyViaETH() function, there is a double validation of whether the totalSupply will be less than or equal to the value of MAX_SWDM_TOKENS. This increases the gas fee costs of the contract both for the moment of deploying it or for making a call of the function.

Recommendation

One possible solution is delete: require(totalSupply() < MAX_SWDM_TOKENS,"All SWDM Tokens Have been minted.");

Disclaimer

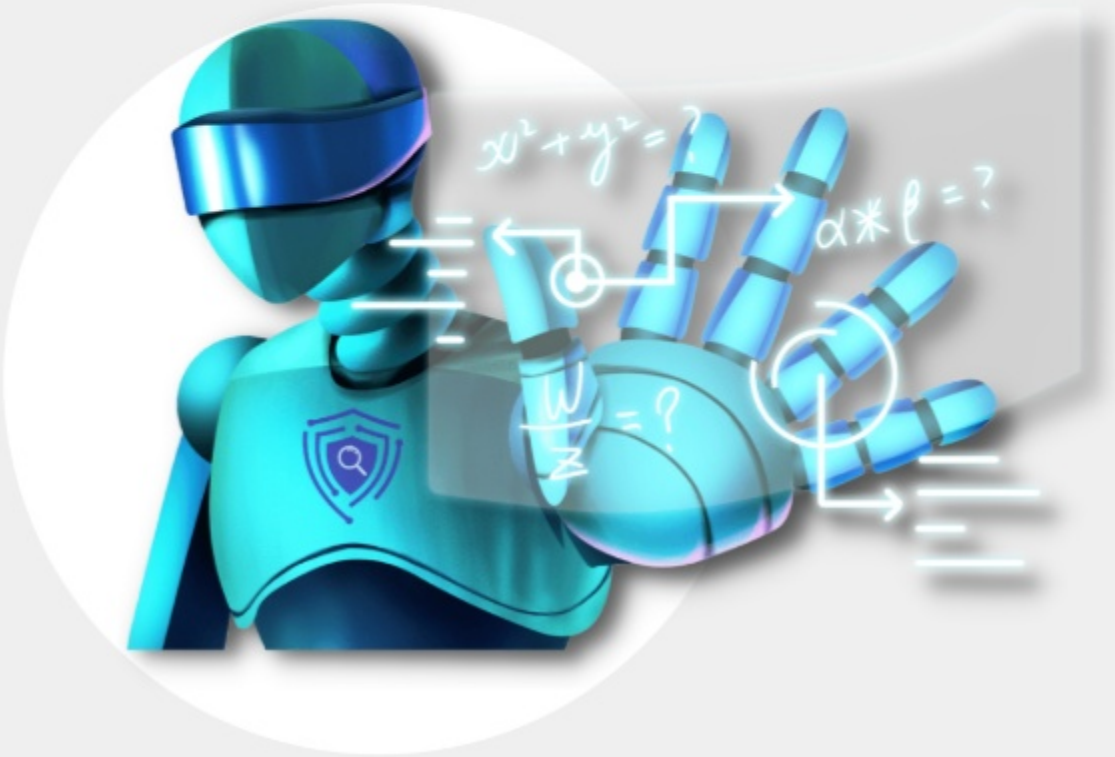
SafuAudit.com is not a financial institution and the information provided on this website does not constitute investment advice, financial advice, trading advice, or any other sort of advice. You should not treat any of the website's content as such. Investing in crypto assets carries a high level of risk and does not hold guarantees for not sustaining financial loss due to their volatility.

Accuracy of Information
SafuAudit will strive to ensure the accuracy of the information listed on this website although it will not hold any responsibility for any missing or wrong information. SafuAudit provides all information as is. You understand that you are using any and all information available here at your own risk. Any use or reliance on our content and services is solely at your own risk and discretion.

The purpose of the audit is to analyze the on-chain smart contract source code and to provide a basic overview of the project.

While we have used all the information available to us for this straightforward investigation, you should not rely on this report only — we recommend proceeding with several independent audits Be aware that smart contracts deployed on a blockchain aren't secured enough against external vulnerability or a hack. Be aware that active smart contract owner privileges constitute an elevated impact on the smart contract safety and security. Therefore, SafuAudit does not guarantee the explicit security of the audited smart contract. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

SAFUAUDIT
SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY



"Only in growth, reform, and change, paradoxically enough, is true security to be found."