YFDFI_FINANCE (YFD)

Smart Contract Partial Audit —
Function_Mint And Constructor Audit

NASDAX.CO
Auditor: Nasdax Security Team
By Lowell Katz (U.S) & Tomer Breteau (U.K)
For Portal By Nasdax

SMART CONTRACT PARTIAL AUDIT REPORT

Security And Fairness Status

SAFE / FAIR

★ ★ ★ ★ ★

CONTRACT NAME : YfDFI_finance

CONTRACT ADDRESS : 0x4F4F0Ef7978737ce928BFF395529161b44e27ad9

01.INTRODUCTION

The Partial Audit is valid for 0x4F4F0Ef7978737ce928BFF395529161b44e27ad9 Ethereum smart contract. YfDFI.finance asked us to review the "Constructor function" and "Function_Mint" in their YFD token smart contract (0x4F4F0Ef7978737ce928BFF395529161b44e27ad9).

Nasdax Security Team reviewed the YfDFI_finance (Contract Name) smart contract and YFD Token deployed on Ethereum Network Mainnet on Block number 11042811 by Oct-12-2020 08:03:39 PM +UTC.
(Txn Hash: 0xcb62f74708a6b3071d24ae1d4e16c3c6c28c2da3d05e421a0dd93902ea6b5d75).

The audit is valid for 0x4F4F0Ef7978737ce928BFF395529161b44e27ad9 contract address only.

02.AUDIT SUMMARY

The YfDFI.finance (YFD) is a Non-Mintable ERC-20 Token. The smart contract code is clean, fair and safe. YFD is based on the latest version of ERC-20 source code standard and strictly following the official guidelines from OpenZeppelin source code provider. The visibility and state mutability of all the functions are clearly specified, and there is no confusion.

No new token can be minted. The 0x4F4F0Ef7978737ce928BFF395529161b44e27ad9 YfDFI_finance Maximum supply is 20,000 YFD and Always will be.

Audit Result : PASSED (SAFE AND FAIR)
Level of "_Mint" Risk : PASSED (NO NEW TOKEN CAN BE MINTED)

03.AUDIT METHODOLOGY

Nasdax Security Team manually reviewed Constructor and _Mint function of the YFD smart contract, following industry best practices and looking for any potential back doors due to Function_Mint.

Nasdax Security Team also used the military level-grade automated process for analysing and reviewing these both function in the YFD smart contract. These tools and automated script looking for any internal or external possibilities to Mint after Construction process. Our process simulate an unfair use of Function_Mint even if it can be used once by Constructor function during the contract deployment only.

04.WHAT YOU CAN FIND IN YFDFI_FINANCE SMART CONTRACT?

YfDFI_finance smart contract uses a constructor combined with a temp Mint function which can be used once for the unique contract deployment.
When a contract is created, its constructor (a function declared with the constructor keyword) is executed once.

These values are immutable: they can only be set once during construction.
It became impossible to call these functions after contract deployment and total supply construction, 20,000 YFD Total is the maximum supply and always will be.

Line 306 to 313 / **RESULT :** APPROVED

```
constructor (string memory name, string memory symbol) public {
        _name = name;
        _symbol = symbol;
        _decimals = 18;
        _mint(msg.sender, 20000000000000000000000);
    }
```

Line 482 to line 490. / **RESULT :** APPROVED

```
function _mint(address account, uint256 amount) internal virtual
{       require(account != address(0), "ERC20: mint to the zero address");
_beforeTokenTransfer(address(0), account, amount);
        _totalSupply = _totalSupply.add(amount);
        _balances[account] = _balances[account].add(amount);
        emit Transfer(address(0), account, amount);
    }
```

Function_Mint is mandatory to mint token supply. Executed with and from constructor, it allows to Mint once only.

05. WHY DO NOT BURN ADMIN KEY AS MINTER AND GOVERNANCE OWNER?

There is No Minter_Role ou Burner_Role in YFD Smart Contract. There is No setGovernance function.

setGovernance function does not exist in YFD contract and can’t be called by input data transaction or other way because it doesn’t exist.

Minter_Role and Burner_Role do not exist in YFD smart contract and can’t be called by input data transaction or other way because it doesn’t exist.

Few smart contracts have this function, that’s why they have to send to their smart contract address these following “Input Data” transactions to burn or remove them:

Function: setGovernance(address _governance)

MethodID: 0xab033ea9
[0]:
000000000000000000000000000000000000000000000000000000000000d
ead

Function: removeMinter(address _minter)

MethodID: 0x3092afd5
[0]:
0000000000000000000000000C09CD04BBF82eD854e4feC7eB10d470f08264
87

In YfDFI_finance smart contract case, Minter_Role, setGovernance and Mint function after construction do not exist in their smart contract. It is impossible to interact with these functions because they do not exist. No need to remove no existing functions.

RESULT : APPROVED, SAFE AND FAIR

06. HOW BEGINNER CAN EASILY VERIFY IT?

To verify all available functions, everyone can use “write contract” interface on Etherscan. This is the easier way to verify which function YfDFI_finance 0x4F4F0Ef7978737ce928BFF395529161b44e27ad9 contract can call post deployment.

https://etherscan.io/address/0x4F4F0Ef7978737ce928BFF395529161b44e27ad9#writeContract

1. approuve (Mandatory)
2. decreaseAllowance (Mandatory) I 3. increaseAllowance (Mandatory)
4. transfer (Mandatory) I 5. transferFrom (Mandatory)

YfDFI_finance smart contract has No back door and No post deployment Mint function.

RESULT : APPROVED, SAFE AND FAIR

07. FINAL RESULT

The YfDFI_finance Maximum supply is 20,000 YFD and it cannot change. Constructor have been allowed once to deploy the smart contract but its keywords cannot be used to Mint, Burn or Govern the YfDFI_Finance supply after construction anymore. There is no Minter Role or Burner Role and constructor keyword is considered as unusable to mint, burn or govern the YfDFI_finance’s smart contract. No new token can be minted. The YfDFI_finance Maximum supply is 20,000 YFD and it will never change.

Combine “Constructor” and “Function_Mint”, and delete Minter_Role/ Burner_Role and setGovernance function to deploy a Non-Mintable token is a good, fair and safe practice but to help people to read and understand that your Total supply is your Maximum supply; it’s always better and clearer to write function_Mint code like that:

```
function _mint(address account, uint256 amount) public onlyOwner {
        require(account != address(0), "ERC20: mint to the zero address");

        require( 20000000000000000000000 >= _totalSupply.safeAdd(amount),
"Max supply 20000 with 18 decimals that is 20000000000000000000000");
        _totalSupply = _totalSupply.safeAdd(amount);
        _balances[account] = _balances[account].safeAdd(amount);
        emit Transfer(address(0), account, amount);
    }
```

08. PURPOSE OF THE REPORT

The Audits and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of Solidity code and only the Solidity code we note as being within the scope of our review within this report. The Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the Solidity programming language that could present security risks. Cryptographic tokens and smart contracts are emergent technologies and carry with them high levels of technical risk and uncertainty.
The Audits are not an endorsement or indictment of any particular project or team, and the Audits do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect, including regarding the bug-free nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Audits in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. This Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. There is no owed duty to any Third-Party by virtue of publishing these Audits.

09. DISCLAIMER

The audit makes no statement or warranty about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statement about reliability of the contracts to purpose, or their “Safety” and “Fairness” status. The audit documentation is for discussion purposes only.

The Content Of This Audit Report Is Provided “As Is”, Without Representation And Warranty Of Any Kind, And Nasdax; Nasdax Security Team Or Portal By Nasdax Disclaims Any Liability For Damage Arising Out Of, Or In Connection With, This Audit Report. Copyright Of This Report Remains With Nasdax Company.

NASDAX LIMITED. PRIVATE LIMITED COMPANY
nasdax
3rd Floor 86-90 Paul Street,
London, United Kingdom, EC2A 4NE
Tel: +44 20 3287 3434
Email: office@nasdax.co
Website: nasdax.co
NASDAX LIMITED. PRIVATE LIMITED COMPANY