



# Security Assessment **TON DEGEN**

Vital Block Security **Verified** on April 19<sup>TH</sup>, 2024

 @Vital-Block

 @VB\_Audit

 info@vitalblock.org

 www.vitalblock.org



PREPARED FOR:  
**TON DEGEN**





# TABLE OF CONTENTS

TABLE OF CONTENTS	3
DOCUMENT PROPERTIES	4
ABOUT VBS	5
SCOPE OF WORK	6
AUDIT METHODOLOGY	7
AUDIT CHECKLIST	9
EXECUTIVE SUMMARY	10
CENTRALIZED PRIVILEGES	11
RISK CATEGORIES	12
AUDIT SCOPE	13
AUTOMATED ANALYSIS	14
KEY FINDINGS	19
MANUAL REVIEW	20
VULNERABILITY SCAN	28
REPOSITORY	29
INHERITANCE GRAPH	30
PROJECT BASIC KNOWLEDGE	31
AUDIT RESULT	32
REFERENCES	37



## INTRODUCTION

<b>Auditing Firm</b>	 <b>VITAL BLOCK SECURITY</b>
<b>Client Firm</b>	 <b>TON DEGEN</b>
<b>Methodology</b>	Automated Analysis, Manual Code Review
<b>Contract</b>	<a href="#">EQCDRnvhhEvZA_rKyI_A_ukFxKy9Rs2Qhc2MHV1Z02MbEAcr</a>
<b>Source Code Light</b>	<b>Verified</b>
<b>Command</b>	<code>func -o output.fif -SPA jetton_minter_discoverable.fc jetton-utils/discovery-params.fc jetton-utils/op.fc jetton-utils/stdlib.fc jetton-utils/utls.fc</code>
<b>Centralization</b>	Contract Renounced
<b>Compiler</b>	FunC
<b>Version</b>	0.4.3
<b>Blockchain</b>	 <b>TON NETWORK</b>
<b>Website</b>	<a href="https://tondegen.com/">https://tondegen.com/</a>
<b>Telegram Group</b>	<a href="https://t.me/ton_degen_en">https://t.me/ton_degen_en</a>
<b>Twitter</b>	<a href="https://twitter.com/Ton_Degen">https://twitter.com/Ton_Degen</a>
<b>White-Paper</b>	<a href="https://tondegen.gitbook.io/ton-degen-documentation">https://tondegen.gitbook.io/ton-degen-documentation</a>
<b>Prelim Report Date</b>	April 18 <sup>th</sup> 2024
<b>Final Report Date</b>	April 19 <sup>th</sup> 2024

 Verify the authenticity of this report on our GitHub Repo: <https://www.github.com/vital-block>



## Document Properties


<b>Client</b>	TON DEGEN
<b>Title</b>	Smart Contract Audit Report
<b>Target</b>	TON DEGEN
<b>Audit Version</b>	1.0
<b>Author</b>	Akhmetshin Marat
<b>Auditors</b>	Akhmetshin Marat, James BK, Benny Matin
<b>Reviewed by</b>	Dima Meru
<b>Approved by</b>	Prince Mitchell
<b>Classification</b>	Public

## Version Info

Version	Date	Author(s)	Description
1.0	April 19 <sup>th</sup> , 2024	James BK	Final Released
1.0-AP	April 19 <sup>th</sup> , 2024	Benny Matin	Release Candidate

## Contact

For more information about this document and its contents, please contact Vital Block Security Inc.

<b>Name</b>	Akhmetshin Marat
<b>Phone</b> 	+44 7944 248057
<b>Email</b>	info@vitalblock.org

In the following, we show the specific pull request and the commit hash value used in this audit.

- <https://tonscan.org/jetton/EQAjYIDrsOmeuDtuPWyTiZ7nPGdSUAt319OzMB6CIk-yhLmz#source> (UTN221761)
- <https://tonviewer.com/EQAjYIDrsOmeuDtuPWyTiZ7nPGdSUAt319OzMB6CIk-yhLmz> (UTN887790)

## About Vital Block Security

Vital Block Security provides professional, thorough, fast, and easy-to-understand smart contract security audit. We do in-depth and penetrative static, manual, automated, and intelligent analysis of the smart contract. Some of our automated scans include tools like ConsenSys MythX, Mythril, Slither, Surya. We can audit custom smart contracts, DApps, Rust, NFTs, etc (including the service of smart contract auditing). We are reachable at Telegram ([https://t.me/vital\\_block](https://t.me/vital_block)), Twitter ([http://twitter.com/Vb\\_Audit](http://twitter.com/Vb_Audit)), or Email ([info@vitalblock.org](mailto:info@vitalblock.org)).

Table 1.2: Vulnerability Severity Classification

Impact	High	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Low
		High	Medium	Low
		Likelihood		

## Methodology (1)

To standardize the evaluation, we define the following terminology based on the OWASP Risk Rating Methodology [4]:

- Likelihood represents how likely a particular vulnerability is to be uncovered and exploited in the wild;
- Impact measures the technical loss and business damage of a successful attack;
- Severity demonstrates the overall criticality of the risk.

## AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit. Here's a brief overview of Vital Block

**Security auditing process and methodology:**

### CONNECT

- **The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.**

### AUDIT

- **Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:**
  - **Remix IDE Developer Tool**
  - **Open Zeppelin Code Analyzer**
  - **SWC Vulnerabilities Registry**
  - **DEX Dependencies, e.g., Pancakeswap, Uniswap**
- **Simulations are performed to identify centralized exploits causing contract and/or trade locks.**
- **A manual line-by-line analysis is performed to identify contract issues and centralized privileges.**

**We may inspect below mentioned common contract vulnerabilities, and centralized exploits:**

<b>Centralized Exploits</b>	<ul style="list-style-type: none"><li>○ <b>Token Supply Manipulation</b></li><li>○ <b>Access Control and Authorization</b></li><li>○ <b>Assets Manipulation</b></li><li>○ <b>Ownership Control</b></li><li>○ <b>Liquidity Access</b></li><li>○ <b>Stop and Pause Trading</b></li><li>○ <b>Ownable Library Verification</b></li></ul>
-----------------------------	--



### Common Contract Vulnerabilities

- Integer Overflow
- Lack of Arbitrary limits
- Incorrect Inheritance Order
- Typographical Errors
- Requirement Violation
- Gas Optimization
- Coding Style Violations
- Re-entrancy
- Third-Party Dependencies
- Potential Sandwich Attacks
- Irrelevant Codes
- Divide before multiply
- Conformance to Solidity Naming Guides
- Compiler Specific Warnings
- Language Specific Warnings

### REPORT

- The auditing team provides a preliminary report specifying all the checks which have been performed and the findings thereof.
- The client's development team reviews the report and makes amendments to the codes.
- The auditing team provides the final comprehensive report with open and unresolved issues.

### PUBLISH

- The client may use the audit report internally or disclose it publicly.

 It is important to note that there is no pass or fail in the audit, it is recommended to view the audit as an unbiased assessment of the safety of solidity codes.



## SCOPE OF WORK

Vital Block was consulted by **TON DEGEN** to conduct the smart contract audit of its. FunC source code. The audit scope of work is strictly limited to mentioned .FunC file only:

0.TONDEGEN.FunC

 External contracts and/or interfaces dependencies are not checked due to being out of scope.

Verify audited contract's contract address and deployed link below:

Public Contract Address	
<a href="https://tonviewer.com/EQCrxBb3eDbZcp51CJDweYQgmi2fgkd11f-jIV3rbnHJgdZz/jetton/EQAjYIDrsOmeuDtuPWyTiZ7nPGdSUAf319OzMB6Clk-yhLmz?section=code">https://tonviewer.com/EQCrxBb3eDbZcp51CJDweYQgmi2fgkd11f-jIV3rbnHJgdZz/jetton/EQAjYIDrsOmeuDtuPWyTiZ7nPGdSUAf319OzMB6Clk-yhLmz?section=code</a>	
Contract Name	TON DEGEN
Token Symbol	DEGEN
Decimals	9
Total Supply	100,000,000





**Table 1.0 The Full Audit Checklist**

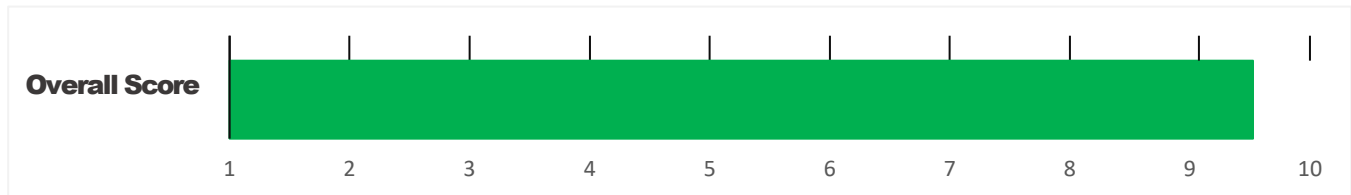
Category	Checklist Items
<b>Basic Coding Bugs</b>	Constructor Mismatch
	Ownership Takeover
	Redundant Fallback Function
	Overflows & Underflows
	Reentrancy
	Money-Giving Bug
	Blackhole
	Unauthorized Self-Destruct
	Revert DoS
	Unchecked External Call
	Gasless Send
	Send Instead Of Transfer
	Costly Loop
	(Unsafe) Use Of Untrusted Libraries
	(Unsafe) Use Of Predictable Variables
	Transaction Ordering Dependence
	Deprecated Uses
<b>Semantic Consistency Checks</b>	Semantic Consistency Checks
<b>Advanced DeFi Scrutiny</b>	Business Logics Review
	Functionality Checks
	Authentication Management
	Access Control & Authorization
	Oracle Security
	Digital Asset Escrow
	Kill-Switch Mechanism
	Operation Trails & Event Generation
	ERC20 Idiosyncrasies Handling
	Frontend-Contract Integration
	Deployment Consistency
	Holistic Risk Management
<b>Additional Recommendations</b>	Avoiding Use of Variadic Byte Array
	Using Fixed Compiler Version
	Making Visibility Level Explicit
	Making Type Inference Explicit
	Adhering To Function Declaration Strictly
	Following Other Best Practices

## EXECUTIVE SUMMARY

Vital Block Security has performed the automated and manual analysis of the **TON DEGEN** FunC code. The code was reviewed for common contract vulnerabilities and centralized exploits. Here's a quick audit summary:

Status	Critical ! 🔴	Major " 🟡	Medium # 🟡	Minor \$ 🟢	Unknown % 🟤
Open	0	0	0	0	2
Acknowledged	0	0	0	1	0
Resolved	0	0	0	0	1
Noteworthy <b>OnlyOwner</b> Privileges	Set Taxes and Ratios, Airdrop, Set Protection Settings, Set Reward Properties, Set Reflector Settings, Set Swap Settings, Set Pair and Router				

**TON DEGEN** Smart contract has achieved the following score: **94.0**



**i** Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.

**i** Please note that centralization privileges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.



## CENTRALIZED PRIVILEGES

**Centralization risk is the most common cause of cryptography asset loss. When a smart contract has a privileged role, the risk related to centralization is elevated.**

**There are some well-intended reasons have privileged roles, such as:**

- **Privileged roles can be granted the power to `pause()` the contract in case of an external attack.**
- **Privileged roles can use functions like, `include()`, and `exclude()` to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale and to list on an exchange.**

**Authorizing privileged roles to externally-owned-account (EOA) is dangerous. Lately, centralization-related losses are increasing in frequency and magnitude.**

- **The client can lower centralization-related risks by implementing below mentioned practices:**
- **Privileged role's private key must be carefully secured to avoid any potential hack.**
- **Privileged role should be shared by multi-signature (multi-sig) wallets.**
- **Authorized privilege can be locked in a contract, user voting, or community DAO can be introduced to unlock the privilege.**
- **Renouncing the contract ownership, and privileged roles.**
- **Remove functions with elevated centralization risk.**

** Understand the project's initial asset distribution. Assets in the liquidity pair should be locked.**

**Assets outside the liquidity pair should be locked with a release schedule.**



## RISK CATEGORIES

Smart contracts are generally designed to hold, approve, and transfer tokens. This makes them very tempting attack targets. A successful external attack may allow the external attacker to directly exploit. A successful centralization-related exploit may allow the privileged role to directly exploit. All risks which are identified in the audit report are categorized here for the reader to review:

Risk Type	Definition
<b>Critical</b> 🚫	These risks could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
<b>Major</b> 🟡	These risks are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.
<b>Medium</b> 🟠	These risks should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. Low-risk re-entrancy-related vulnerabilities should be fixed to deter exploits.
<b>Minor</b> 🟢	These risks do not pose a considerable risk to the contract or those who interact with it. They are code-style violations and deviations from standard practices. They should be highlighted and fixed nonetheless.
<b>Unknown</b> 🟤	These risks pose uncertain severity to the contract or those who interact with it. They should be fixed immediately to mitigate the risk uncertainty.

All statuses which are identified in the audit report are categorized here for the reader to review:

Status Type	Definition
<b>Open</b>	Risks are open.
<b>Acknowledged</b>	Risks are acknowledged, but not fixed.
<b>Resolved</b>	Risks are acknowledged and fixed.



## Key Findings






Overall, these contracts are well-designed and engineered, though the implementation can be improved by resolving the identified issues (shown in Table [2.1](#)), 0 medium-severity vulnerabilities, 3 low-severity vulnerabilities, and 1 informational recommendations.

Table 2.1: Key **TON DEGEN** Audit Findings

ID	Severity	Title	Category	Status
UTN-01	Low	<a href="#">In updateForMinter, the following equation is used inside an unchecked block</a>	Coding Practice	Acknowledged

Beside the identified issues, we emphasize that for any user-facing applications and services, it is always important to develop necessary risk-control mechanisms and make contingency plans, which may need to be exercised before the mainnet deployment. The risk-control mechanisms should kick in at the very moment when the contracts are being deployed on mainnet. Please refer to page [10](#) for details...

## AUTOMATED ANALYSIS

Symbol	Definition
	Function modifies state
	Function is payable
	Function is internal
	Function is private
	Function is important

```

| **TON DEGEN** | Interface | |||NO! |
(int, slice, cell, cell) load_data() inline {
    slice ds = get_data().begin_parse(); |NO! |
    return (
        ds~load_coins(), ;; total_supply |NO! |
        ds~load_msg_addr(), ;; admin_address |NO! |
        ds~load_ref(), ;; content ! |NO! |
        ds~load_ref() ;; jetton_wallet_code |NO! |
    );
}
;; storage scheme |NO! |
;; storage#_total_supply:Coins |NO! |
admin_address:MsgAddress content:^Cell |NO! |
jetton_wallet_code:^Cell = Storage; |NO! |

(int, slice, cell, cell) load_data() inline { |NO! |
    slice ds = get_data().begin_parse(); |NO! |
    return (
        ds~load_coins(), ;; total_supply |NO! |
        ds~load_msg_addr(), ;; admin_address |NO! |
        ds~load_ref(), ;; content |NO! |
        ds~load_ref() ;; jetton_wallet_code |NO! |
    );
}

() save_data(int total_supply, slice
admin_address, cell content, cell jetton_wallet_c |NO! |
impure inline {
    set_data(begin_cell() |NO! |
        .store_coins(total_supply)
        .store_slice(admin_address) ! |NO! |
        .store_ref(content)
        .store_ref(jetton_wallet_code) |NO! |
        .end_cell());
}

() mint_tokens(slice to_address, cell |NO! |
jetton_wallet_code, int amount, cell master_msg) |NO! |
{
    cell state_init =
calculate_jetton_wallet_state_init(to_address ,
my_address(), jetton_wallet_code);

```



```

slice to_wallet_address =
calculate_jetton_wallet_address(state_init); |NO! |
    var msg = begin_cell() |NO! |
        .store_uint(0x18, 6) |NO! |
        .store_slice(to_wallet_address) |NO! |
        .store_coins(amount) |NO! |
        .store_uint(4 + 2 + 1, 1 + 4 + 4 + 64 + 32 + 1 + 1 + 1) |NO! |
        .store_ref(state_init) |NO! |
        .store_ref(master_msg) |NO! |
    ;
    send_raw_message(msg.end_cell(), 1); ;; pay transfer fees |NO! |
separately, revert on errors |NO! |
}

() rcv_internal(int msg_value, cell in_msg_full, slice
in_msg_body) impure { |NO! |
    if (in_msg_body.slice_empty?()) { ;; ignore empty
messages |NO! |
        return () |NO! |
    ;
} |NO! |
    slice cs = in_msg_full.begin_parse() |NO! |
    ;
    int flags = cs~load_uint(4); |NO! |

    if (flags & 1) { ;; ignore all bounced messages |NO! |
        return (); |NO! |
    }

    () rcv_internal(int msg_value, cell in_msg_full, slice |NO! |
in_msg_body) impure { |NO! |
        if (in_msg_body.slice_empty?()) { ;; ignore empty
messages |NO! |
            return (); |NO! |
        }
        slice cs = in_msg_full.begin_parse(); |NO! |
        int flags = cs~load_uint(4); |NO! |

        if (flags & 1) { ;; ignore all bounced messages |NO! |
            return (); |NO! |
        }
        slice sender_address = cs~load_msg_addr(); |NO! |
        cs~load_msg_addr(); ;; skip dst |NO! |
        cs~load_coins(); ;; skip value |NO! |
        cs~skip_bits(1); ;; skip extracurrency collection |NO! |
        cs~load_coins(); ;; skip ihr_fee |NO! |
        int fwd_fee = cs~load_coins(); ;; we use message fwd_fee |NO! |
for estimation of provide_wallet_address cost |NO! |

        int op = in_msg_body~load_uint(32); |NO! |
        int query_id = in_msg_body~load_uint(64); |NO! |

        (int total_supply, slice admin_address, cell content, cell
jetton_wallet_code) = loa

```

```

mint_tokens(to_address, jetton_wallet_code, amount, |NO!|
master_msg);
    save_data(total_supply + jetton_amount, admin_address, |NO!|
content, jetton_wallet_code);
    return (); ! ● |NO!|
} |NO!|

if (op == op::burn_notification()) { |NO!|
    int jetton_amount = in_msg_body~load_coins();
    slice from_address = in_msg_body~load_msg_addr(); ● |NO!|

    throw_unless(74, ! ● |NO!|
equal_slices(calculate_user_jetton_wallet_address(from_address, |NO!|
my_address(), jetton_wallet_code), sender_address) |NO!|
);

    save_data(total_supply - jetton_amount, admin_address, |NO!|
content, jetton_wallet_code);
    slice response_address = in_msg_body~load_msg_addr() |NO!|
; |NO!|

    if (response_address.preload_uint(2) != 0) |NO!|
    {
        var msg = begin_cell() |NO!|
        .store_uint(0x10, 6) ;; nobounce - int_msg_info$0
ihr_disabled:Bool bounce:Bool bounced:Bool src:MsgAddress -> ! ● |NO!|
011000
        .store_slice(response_address) |NO!|
        .store_coins(0) |NO!|
        .store_uint(0, 1 + 4 + 4 + 64 + 32 + 1 + 1)
        .store_uint(op::excesses(), 32) |NO!|
        .store_uint(query_id, 64);
        send_raw_message(msg.end_cell(), 2 + 64) ! ● |NO!|
; |NO!|
    }

    return (); |NO!|
} |NO!|

if (op == op::provide_wallet_address()) { |NO!|
    throw_unless(75, msg_value > fwd_fee +
provide_address_gas_consumption()); |NO!|

    slice owner_address = in_msg_body~load_msg_addr(); |NO!|
    int include_address? = in_msg_body~load_uint(1); |NO!|

    cell included_address = include_address?
? begin_cell().store_slice(owner_address).end_cell()
: null(); |NO!|

| L | setExcludedFromFees |NO!|

```



## TDG-01 POSSIBLE OVERFLOW

Category	Severity ●	Location	Status
Status Mathematical Operations	Minor	Contract/code/TONDEGEN #70-81	Acknowledged

### Description

In `updateForMinter`, the following equation is used inside an unchecked block

```
if (op == op::mint()) { throw unless(73, equal_slices(sender_address, admin_address)); slice to address =
in_msg_body~load_msg_addr(); int amount = in_msg_body~load_coins(); cell master_msg = in_msg_body~load_ref(); slice
master_msg_cs = master_msg.begin_parse(); master_msg_cs~skip_bits(32 + 64); ;; op + query id int jetton amount =
master_msg_cs~load_coins(); mint_tokens(to_address, jetton_wallet_code, amount, master_msg); save_data(total_supply +
jetton_amount, admin_address, content, jetton_wallet_code); return (); }
```






Minter can not issue more **DEGEN** tokens indefinitely.

Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the **DEGEN** contract.

### Recommendation

We recommend either checking for overflow in this case, or ensuring that the **PairsIn** is close enough it will never cause an overflow.

## OPTIMIZATIONS | TON DEGEN

ID	Title	Category	Status
FTV	Logarithm Refinement Optimization	Gas Optimization	Acknowledged 
FOP	Checks Can Be Performed Earlier	Gas Optimization	Acknowledged 
FDP	Unnecessary Use Of SafeMath	Gas Optimization	Acknowledged 
FWY	Struct Optimization	Gas Optimization	Acknowledged 
FGT	Unused State Variable	Gas Optimization	Acknowledged 

## General Detectors

### Missing Zero Address Validation

Some functions in this contract may not appropriately check for zero addresses being used.









































Attention  
Required

### Uninitialized Local Variables

This contract's local variables are not all initialized, potentially resulting in lost funds or other exploits.



Attention  
Required

- |  |  |
|--|--|
|  No compiler version inconsistencies found      |  No tautologies or contradictions found                       |
|  No unchecked call responses found              |  No faulty true/false values found                            |
|  No vulnerable self-destruct functions found    |  No inaccurate divisions found                                |
|  No assertion vulnerabilities found            |  No redundant constructor calls found                        |
|  No old solidity code found                   |  No vulnerable transfers found                              |
|  No external delegated calls found            |  No vulnerable return values found                          |
|  No external call dependency found            |  No uninitialized local variables found                     |
|  No vulnerable authentication calls found     |  No default function responses found                        |
|  No invalid character typos found             |  No missing arithmetic events found                         |
|  No RTL characters found                      |  No missing access control events found                     |
|  No dead code found                           |  No redundant true/false comparisons found                  |
|  No risky data allocation found               |  No state variables vulnerable through function calls found |
|  No uninitialized state variables found       |  No buggy low-level calls found                             |
|  No uninitialized storage variables found     |  No expensive loops found                                   |
|  No vulnerable initialization functions found |  No bad numeric notation practices found                    |
|  No risky data handling found                 |  No missing constant declarations found                     |
|  No number accuracy bug found                 |  No missing external function declarations found            |
|  No out-of-range number vulnerability found   |  No vulnerable payable functions found                      |
|  No map data deletion vulnerabilities found   |  No vulnerable message values found                         |



## Vulnerability Scan

### REENTRANCY

✓ No reentrancy risk found

Severity	Minor
Confidence Parameter	Certain

✗ **Not Mintable:** A large amount of this token can not be minted by a private wallet or contract.

## Vulnerability Description

## Scanning Line:

```

if (op == op::burn_notification()) {
    int jetton_amount = in_msg_body~load_coins();
    slice from_address = in_msg_body~load_msg_addr();
    throw_unless(74,

equal_slices(calculate_user_jetton_wallet_address(from_address
, my_address(), jetton_wallet_code), sender_address)
);
    save_data(total_supply - jetton_amount, admin_address,
content, jetton_wallet_code);
    slice response_address = in_msg_body~load_msg_addr();
    if (response_address.preload_uint(2) != 0) {
        var msg = begin_cell()
            .store_uint(0x10, 6) ;; nobounce -
int_msg_info$0 ihr_disabled:Bool bounce:Bool bounced:Bool
src:MsgAddress -> 011000
            .store_slice(response_address)
            .store_coins(0)
            .store_uint(0, 1 + 4 + 4 + 64 + 32 + 1 +
1)
            .store_uint(op::excesses(), 32)
            .store_uint(query_id, 64);
        send_raw_message(msg.end_cell(), 2 + 64);
    }
}

```

Identifier	Definition	Severity
CEN-02	Initial asset distribution	Minor 

```
if (op == op::provide_wallet_address()) {
    throw_unless(75, msg_value > fwd_fee +
const::provide_address_gas_consumption());

    slice owner_address = in_msg_body~load_msg_addr();
    int include_address? = in_msg_body~load_uint(1);

    cell included_address = include_address?
        ?
begin_cell().store_slice(owner_address).end_cell()
    : null();
```

## Description:

Floating point calculations can vary across different architectures.

## Alleviation:

This exhibit was acknowledged and ultimately discarded by the **TON DEGEN** team due to low severity. We consider the exhibit fully attended to as it doesn't impose any meaningful security concerns.

## RECOMMENDATION

**Project stakeholders should be consulted during the initial asset distribution process.**



## Contract Owner Address:

<https://tonscan.org/address/UQAAAJKZ>

## Audited Files

TONDEGEN.Func

## Contracts:

Contract

DEGEN::EQAjYlDrsOmeuDtuPWyTiZ7nPGdSUAt3190zMB6CIk-yhLmz

## Creator TXN Hash:

Txn::e4AU7B1/0CQ02IpI+LsXpCrhMHWGZ+3KGyb2MEzA2s8=



## Vulnerability Run check

### Risk Analysis

#### ✓ Contract source code verified

This token contract is open source. You can check the contract code for details. Unsourced token contracts are likely to have malicious functions to defraud their users of their assets.

#### ✓ No mint function

Mint function is transparent or non-existent. Hidden mint functions may increase the amount of tokens in circulation and effect the price of the token.

#### ✓ Owner cant change balance

The contract owner does not have the authority to modify the balance of tokens at other addresses.

#### ✓ No Proxy

There is no proxy in the contract. The proxy contract means contract owner can modify the function of the token and possibly effect the price.

#### ✓ No function to retrieve ownership

If this function exists, it is possible for the project owner to regain ownership even after relinquishing it.



### Honeypot Risk

#### ✓ This does not appear to be a honeypot

We are not aware of any code that prevents the sale of tokens.

#### ✓ No Anti Whale

There is no limit to the number of token transactions. The number of scam token transactions may be limited (honeypot risk).

#### ✓ No whitelist function

Whitelist function found

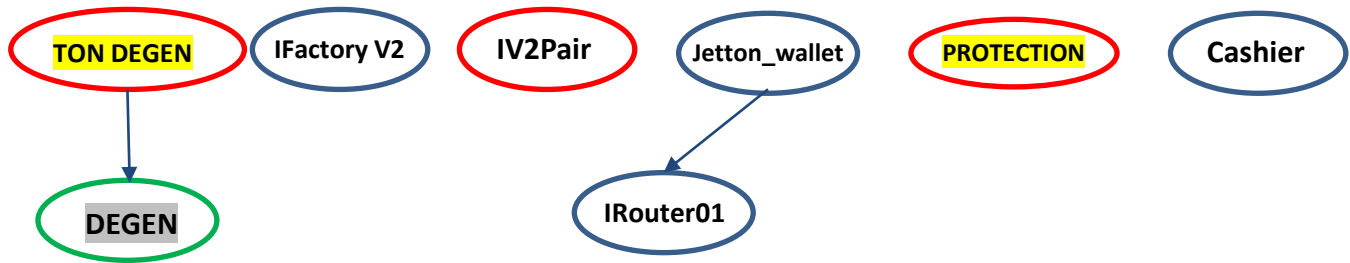
#### ✓ No trading cooldown

The token contract has no trading cooldown function. If there is a trading cooldown function, the user will not be able to sell the token within a certain time or block after buying.

#### ✓ No blacklist function

No blacklist function is included.

## INHERITANCE GRAPH



Identifier	Definition	Severity
CEN-12	Centralization privileges of <b>TON DEGEN</b>	Medium # 🟡

Vulnerability 0 : No important security issue detected.

Threat level: **Low**

```

20  ds~load_ref(), ;; content
21  ds~load_ref() ;; jetton_wallet_code
22  );
23  }
24
25  () save_data(int total_supply, slice admin_address, cell content, cell jetton_wallet_code) impure inline {
26      set_data(begin_cell()
27          .store_coins(total_supply)
28          .store_slice(admin_address)
29          .store_ref(content)
30          .store_ref(jetton_wallet_code)
31          .end_cell()
32      );
33  }
34
35  () mint_tokens(slice to_address, cell jetton_wallet_code, int amount, cell master_msg) impure {
36      cell state_init = calculate_jetton_wallet_state_init(to_address, my_address(), jetton_wallet_code);
37      slice to_wallet_address = calculate_jetton_wallet_address(state_init);
38      var msg = begin_cell()
39          .store_uint(0x18, 6)
40          .store_slice(to_wallet_address)
41          .store_coins(amount)
42          .store_uint(4 + 2 + 1, 1 + 4 + 4 + 64 + 32 + 1 + 1 + 1)
43          .store_ref(state_init)
  
```



## MANUAL REVIEW

**TON DEGEN:** TonDegen is a memecoin that aims to bring value to the community by informing them about TON and the TON ecosystem.

**TOKEN NAME:** TON DEGEN

**Ticker:** DEGEN

**Total Supply:** 100,000,000

**Chain/Standard:** TON Network



The **DEGEN TON** Platform Is Launching On the **TON** Network

[About](#)

[Tokemonics](#)

[Presale](#)

[How to buy](#)

[Docs](#)

[Social](#)

# TON DEGEN

Discover the newest token  
on the TON blockchain

Begin





# ISSUES CHECKING STATUS

Issue Description

Checking Status

1.	Compiler errors.	PASSED
2.	Race Conditions and reentrancy. Cross-Function Race Conditions.	PASSED
3.	Possible Delay In Data Delivery.	PASSED
4.	Oracle calls.	PASSED
5.	Front Running.	PASSED
6.	TON Dependency.	PASSED
7.	Integer Overflow And Underflow.	PASSED
8.	DoS with Revert.	PASSED
9.	Dos With Block Gas Limit.	PASSED
10.	Methods execution permissions.	PASSED
11.	Economy Model of the contract.	PASSED
12.	The Impact Of Exchange Rate On the TON Logic.	PASSED
13.	Private use data leaks.	PASSED
14.	Malicious Event log.	PASSED
15.	Scoping and Declarations.	PASSED
16.	Uninitialized storage pointers.	PASSED
17.	Arithmetic accuracy.	PASSED
18.	Design Logic.	PASSED
19.	Cross-Function race Conditions	PASSED
20.	Save Upon Move contract Implementation and Usage.	PASSED
21.	Fallback Function Security	PASSED



AUDIT RESULT

PASSED

SMART CONTRACT AUDIT OF TON DEGEN

Identifier	Definition	Severity
CEN-02	Initial asset distribution	Minor 

All of the initially minted assets are sent to the contract deployer when deploying the contract. This can be an issue as the deployer and/or contract owner can distribute tokens without consulting the community.

```
if (is_resolvable(owner_address)) {  
    msg = msg.store_slice(calculate_user_jetton_wallet_address(owner_address,  
my_address(), jetton_wallet_code));  
} else {  
    msg = msg.store_uint(0, 2); ;; addr_none  
}  
send_raw_message(msg.store_maybe_ref(included_address).end_cell(), 64);  
return ();
```

## RECOMMENDATION

Project stakeholders should be consulted during the initial asset distribution process.



## RECOMMENDATION

**Deployer and/or contract owner private keys are secured carefully.**

**Please refer to PAGE-09 **CENTRALIZED PRIVILEGES** for a detailed understanding.**

## ALLEVIATION

**The **TON DEGEN** project team understands the centralization risk. Some functions are provided privileged access to ensure a good runtime behavior in the project**



## References

- 1 MITRE. CWE-1041: Use of Redundant Code. <https://cwe.mitre.org/data/definitions/1041.html>.
- 2 MITRE. CWE-1099: Inconsistent Naming Conventions for Identifiers. <https://cwe.mitre.org/data/definitions/1099.html>.
- 3 MITRE. CWE-561: Dead Code. <https://cwe.mitre.org/data/definitions/561.html>.
- 4 MITRE. CWE-563: Assignment to Variable without Use. <https://cwe.mitre.org/data/definitions/563.html>.
- 5 MITRE. CWE-663: Use of a Non-reentrant Function in a Concurrent Context. <https://cwe.mitre.org/data/definitions/663.html>.
- 6 MITRE. CWE-837: Improper Enforcement of a Single, Unique Action. <https://cwe.mitre.org/data/definitions/837.html>.
- 7 MITRE. CWE-841: Improper Enforcement of Behavioral Workflow. <https://cwe.mitre.org/data/definitions/841.html>.
- 8 MITRE. CWE CATEGORY: Bad Coding Practices. <https://cwe.mitre.org/data/definitions/1006.html>.
- 9 MITRE. CWE CATEGORY: Business Logic Errors. <https://cwe.mitre.org/data/definitions/840.html>.
- 10 MITRE. CWE CATEGORY: Concurrency. <https://cwe.mitre.org/data/definitions/557.html>.
- 11 MITRE. CWE VIEW: Development Concepts. <https://cwe.mitre.org/data/definitions/699.html>.
- 12 OWASP. Risk Rating Methodology. [https://www.owasp.org/index.php/OWASP\\_Risk\\_Rating\\_Methodology](https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology).

Identifier	Definition	Severity
COD-10	Third Party Dependencies	Minor 

Smart contract is interacting with third party protocols e.g., Pancakeswap router, cashier contract, protections contract. The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised, and exploited. Moreover, upgrades in third parties can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

## RECOMMENDATION

Inspect and validate third party dependencies regularly, and mitigate severe impacts whenever necessary.



## CERTIFICATE BY VITAL BLOCK SECURITY



## DISCLAIMERS

Vital Block provides the easy-to-understand audit of Solidity, Move and Raw source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high levels of technical risks and uncertainty. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. This audit report could include false positives, false negatives, and other unpredictable results.

## CONFIDENTIALITY

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without InterFi Network's prior written consent.

## NO FINANCIAL ADVICE

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. This audit report should not be used in any way





to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

**FOR AVOIDANCE OF DOUBT, SERVICES, INCLUDING ANY ASSOCIATED AUDIT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.**

### **TECHNICAL DISCLAIMER**

**ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, VITAL BLOCK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO SERVICES, AUDIT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, VITAL BLOCK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.**

**WITHOUT LIMITING THE FOREGOING, VITAL BLOCK MAKES NO WARRANTY OF ANY KIND THAT ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CLIENT’S OR ANY OTHER INDIVIDUAL’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.**

### **TIMELINESS OF CONTENT**

**The content contained in this audit report is subject to change without any prior notice. Vital Block does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.**



## **LINKS TO OTHER WEBSITES**

**This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than Vital Block. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites and social accounts owners. You agree that Vital block Security is not responsible for the content or operation of such websites and social accounts and that Vital Block shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.**



## ABOUT VITAL BLOCK

**Vital Block provides intelligent blockchain Security Solutions. We provide solidity and Raw Code Review, testing, and auditing services. We have Partnered with 25+ Crypto Launchpads, audited 1450+ smart contracts, and analyzed 200,000+ code lines. We have worked on major public blockchains e.g., Ethereum, Binance, Cronos, Doge, Polygon, Avalanche, Metis, Fantom, Bitcoin Cash, Aptos, Oasis, TON, etc.**

**Vital Block is Dedicated to Making Defi & Web3 A Safer Place. We are Powered by Security engineers, developers, UI experts, and blockchain enthusiasts. Our team currently consists of 5 core members, and 4+ casual contributors.**

**Website:** <https://Vitalblock.org> **Email:**

[info@vitalblock.org](mailto:info@vitalblock.org)

**GitHub:** <https://github.com/vital-block>

**Telegram (Engineering):** [https://t.me/vital\\_block](https://t.me/vital_block) **Telegram**

**(Onboarding):** [https://t.me/vitalblock\\_cmo](https://t.me/vitalblock_cmo)





**vital-block**



**info@vitalblock.org**



**www.Vitalblock.org**



Vital Block Dedicated to securing Public and Private Blockchain Ecosystem