

Bool Network Smart Contract Audit Report

Sun Jul 28 2024



contact@bitslab.xyz



https://twitter.com/tonbit_



Bool Network Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	An external verification model to facilitate arbitrary message transmission (AMT) across heterogeneous networks.
Type	Bridge
Auditors	TonBit
Timeline	Tue Jun 25 2024 - Sun Jul 28 2024
Languages	FunC
Platform	Ton
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/boolnetwork/bool-ton-contracts-v1
Commits	f4033a1ef4d6f26d9cdb25a64b2631605550d6451b8c3b66b1cd1a02bb85e6fc5da706b1267fe7ccd10e9befa355b864919b74a6269de022b17364fa6a1fca4cbb5e5e061993fa02062c2c951457e84a6a1fca4cbb5e5e061993fa02062c2c951457e848a960f34d0bdce47b1fee96e8b3927ae2030b258

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
MES	contracts/messenger.fc	e555b7c1e688093a0753bbf21418 ece1a0281f9e
JBR	contracts/jetton-bridge.fc	ccb94ed39b90026ddbb03421dfee 6e2ba934c43b
HE3	contracts/helloweb3.fc	2ecd551606bed185d5d1d8192a87 1aad98966b0e
FEE	contracts/fee.fc	a29e9f8c16fd9b674d74647975e20 f14390b0482
JWA	contracts/swap/jetton-wallet/jetton -wallet.fc	b4f52ecb3fbc20e8899a562ca2752 bba42c12088
PAR	contracts/swap/jetton-wallet/import s/params.fc	3e86ce82bee70992c9b0f7b4fcacf0 cacfcfec1b
STD	contracts/swap/jetton-wallet/import s/stdlib.fc	48ba5be2230d6db462adb890e7b 15ff0b36b90de
OCO	contracts/swap/jetton-wallet/import s/op-codes.fc	de6e2645c68d08535a353fa1b6bd e7ac915d8ef5
UTI	contracts/swap/jetton-wallet/import s/utls.fc	19cd144cd1353e5179c9cefdd1e9b 4f484f4b016
CON	contracts/swap/jetton-wallet/import s/constants.fc	4630656a3a259560d0f4971082975 4698357f4d1
JUT	contracts/swap/jetton-wallet/import s/jetton-utls.fc	e725b3a317c7c347307c6c7a4b68 9119c04c8b58

JMI	contracts/swap/jetton-wallet/jetton-minter.fc	20f6f25543e8c2027c78a8c464fd0acf4617a236
POO	contracts/swap/pool/pool.fc	6e7f1bacd0cb735a5f5dc8e581fc72fa6f100bb6
UTI1	contracts/swap/pool/utis.fc	6236263904b6b55abbaa13814f99d1d6f8507025
ERR	contracts/swap/router/error.fc	947e979d9aa53ff9210f950ec422155d957d18c3
ROU	contracts/swap/router/router.fc	df60d95f5ce097fe92c243d69342523d8198a9ce
PCA	contracts/swap/router/pool-calls.fc	ee685afde74c30e1910a7d0ec6ce0ce6921d9c09
UTI2	contracts/swap/router/utis.fc	a987842e92fd2704238d1b3de6d74982a1876383
BSC	contracts/swap/bool-swap-consumer.fc	3310bf162f3fef3ffd57845ef134ce4114a96865
STD1	contracts/imports/stdlib.fc	2f104cd568a4cebb1c4112ecf8979800f0672575
ANC	contracts/anchor.fc	9b0dca3f244de79c84bb15fe5e0c257cd5cc9381
UTI3	contracts/utis.fc	e1c9ac213361fee2421666b243b29d23bcbb3be7

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	12	12	0
Informational	0	0	0
Minor	6	6	0
Medium	3	3	0
Major	2	2	0
Critical	1	1	0

1.4 TonBit Audit Breakdown

TonBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values

1.5 Methodology

The security team adopted the "**Testing and Automated Analysis**", "**Code Review**" strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by [Bool Network](#) to identify any potential issues and vulnerabilities in the source code of the [Bool Network](#) smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 12 issues of varying severity, listed below.

ID	Title	Severity	Status
MES-1	Lack of Events Emit	Minor	Fixed
MES-2	The <code>enable_global_path</code> function Lacks Permission Validation	Minor	Fixed
PCA-1	Redundant Exception Throwing	Minor	Fixed
POO-1	Emit Forged Message	Critical	Fixed
POO-2	Incorrect Permission of Setting Rates	Major	Fixed
POO-3	Taking Out More Liquidity than Reserve May Result in A Loss of Assets.	Major	Fixed
POO-4	Changing Token Types Causes Asset Errors	Medium	Fixed
POO-5	Incorrect Judgement	Medium	Fixed
POO-6	Lack of Native Token Swap Limit Check	Medium	Fixed
POO-7	Incorrect Exception Throwing	Minor	Fixed

POO-8	Error Code Not Used	Minor	Fixed
POO-9	Calculating Gas Consumption without Checking for Sufficiency	Minor	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the [Bool Network](#) Smart Contract :

Admin

- The `Admin` can enable cross chain path by setting status of the chain id through sending `messenger::enable_global_path` ;
- The `Admin` can update the admin address through sending `messenger::update_admin` .
- The `Admin` can update the code through sending `op::update_code` .
- The `Admin` can update the fee ratio through sending `pool::set_fee_ratio` .
- The `Admin` can register the swap consumer through sending `pool::register_swap_consumer` .
- The `Admin` can update the jetton wallet address through sending `pool::update_jetton_wallet_addr` .
- The `Admin` can update the anchor through sending `pool::update_anchor` .
- The `Admin` can update the swap limit amount through sending `pool::update_swap_limit` .
- The `Fee Admin` can set new fee admin and new fee config through sending `messenger::set_fee_admin` and `messenger::set_fee_config` .
- The `Fee Receiver` can set new fee receiver and withdraw the fee from contract through sending `messenger::set_fee_receiver` and `messenger::withdraw_fee` .
- The `Admin` can update the `ctx_jetton_master` and `ctx_is_locked_jetton` through sending `op::update_binding` .
- The `Admin` can update the `max_import_span` through sending `op::update_max_import_span` .
- The `Admin` can update the `max_unsuccessful_num_limit` through sending `op::update_max_unsuccessful_num_limit` .
- The `Admin` can remove the unsuccessful nonce through sending `op::remove_unsuccessful_nonce` .

User

- The `User` can transfer tokens to bridge contract to do cross-chain operation through sending `op::transfer_notification()` or `pool::swap_in` .
- The `User` can provide liquidity through sending `pool::increase_liquidity` .
- The `User` can withdraw tokens from own position through sending `pool::decrease_liquidity` or `pool::decrease_liquidity_remote` .

4 Findings

MES-1 Lack of Events Emit

Severity: Minor

Status: Fixed

Code Location:

contracts/messenger.fc#178,188,198,375,403;

contracts/swap/pool/pool.fc#591,612,682

Descriptions:

The contract lacks appropriate events for monitoring sensitive operations, which could make it difficult to track sensitive actions or detect potential issues. For example, the `set_fee_admin` , `set_fee_receiver` , `set_fee_config` , `update_consumer` , `enable_path` , and so on.

Suggestion:

It is recommended to emit events for those important functions.

Resolution:

The client adopted the suggestion and fixed this issue.

MES-2 The `enable_global_path` function Lacks Permission Validation

Severity: Minor

Status: Fixed

Code Location:

`contracts/messenger.fc#108`

Descriptions:

Anyone can add a new `chain_id` to the contract, which will consume contract storage space, increase gas costs, and potentially lead to security risks.

Suggestion:

It is recommended to confirm if it aligns with the design.

Resolution:

The client adopted the suggestion and fixed this issue.

PCA-1 Redundant Exception Throwing

Severity: Minor

Status: Fixed

Code Location:

contracts/swap/router/pool-calls.fc#24

Descriptions:

In the case of business processing and release of events based on different topics, the value of the topic for exceptions is tested in line 24 of the `handle_emit_event()` function, but the value of the topic is already limited at the beginning of this if statement, so it is redundant.

Suggestion:

It is recommended to delete this line and make sure it fit with your design.

Resolution:

The client adopted the suggestion and fixed this issue.

POO-1 Emit Forged Message

Severity: Critical

Status: Fixed

Code Location:

contracts/swap/pool/pool.fc

Descriptions:

Firstly, the attacker need to register our Anchor in the A chain messenger and set the

`Consumer` and `Admin` to the attacker address. Then register the `Anchor` in chain B and set `Consumer` to the address of the pool with the asset.

These executions will succeed because `register_anchor` can be called by anyone. Then we call `messenger::send_message`. It passes inspection here because this anchor is set by the attacker.

Now messenger will release the event and Bool Monitor Service will detect this event and send a message to the B chain. On the B chain, the `receive_message` function is executed, and since the message body is all forged by the attacker and the `Anchor` is controlled by the attacker, it is possible to pass the checking of the Anchor correspondence.

Since the private key in the `Anchor` is also controlled by the attacker, it is able to pass the signature checking.

Now take out the consumer in the anchor and send the message constructed by the attacker. This consumer is set up by the attacker after registering the anchor and is a pool with real assets. The messenger then sends a message to the pool, executing `receive_message_from_messenger` in the pool.

And the source of the messenger is only checked in the pool, which may not identify the attacker's forged message. The attacker passes the messenger's check by forging anchors and points one of them to the real pool, thus sending a fake message to manipulate the assets in this pool.

Suggestion:

It is recommended to fix this by checking the anchor mapping relations or other checkings.

Resolution:

The client adopted the suggestion and added the anchor checking to fix this issue.

POO-2 Incorrect Permission of Setting Rates

Severity: Major

Status: Fixed

Code Location:

contracts/swap/pool/pool.fc#435

Descriptions:

When `op == pool::set_fee_ratio`, lack of permission checks when setting handling rates, which allows everyone to modify rates, resulting in pool rates that are too low or too high to function.

Suggestion:

It is recommended to add permission control.

Resolution:

The client adopted the suggestion and fixed this issue.

POO-3 Taking Out More Liquidity than Reserve May Result in A Loss of Assets.

Severity: Major

Status: Fixed

Code Location:

contracts/swap/pool/pool.fc#264

Descriptions:

When `indicator == REMOTE_SWAP_OUT` it removes the specified amount of liquidity. When the removed liquidity is greater than the `reserve` provided by the pool, it will choose to add the amount quantity to the user's position. We know from the return value of the function `handle_remote_remove_liquidity()` that at this point the `exit_code` is 0, and instead of returning the result to the messenger it will continue to execute, changing the position, and then executing to transfer the funds because there is not enough amount in the contract to pay for the transaction, causing an error to be reported, and at this point there is not a transfer to the messenger to send any message, which may lead to asset desynchronisation between the chains, which in turn leads to asset loss.

Suggestion:

It is recommended to make sure this fits your design.

Resolution:

The client adopted the suggestion and fixed this issue.

POO-4 Changing Token Types Causes Asset Errors

Severity: Medium

Status: Fixed

Code Location:

contracts/swap/pool/pool.fc#591

Descriptions:

If the token type is changed when changing the jetton wallet address, this will result in the number of tokens in the original user's position being taken out of the newly changed number of tokens. This is due to the fact that when changing the token type, the position information is still the same as the previous token, which can lead to a serious loss of funds.

Suggestion:

It is recommended to update your jetton wallet address with the same token.

Resolution:

The client adopted the suggestion and fixed this issue.

POO-5 Incorrect Judgement

Severity: Medium

Status: Fixed

Code Location:

contracts/swap/pool/pool.fc#848

Descriptions:

In the `handle_swap_out` function, when `adjusted_amount` and `fee` are judged, `adjusted_amount` has already deducted the `fee`, and further judgment will result in an incorrect result being returned.

```
adjusted_amount -= fee;
if (adjusted_amount < fee) {
    exit_code = error::insufficient_fee;
    return (exit_code, 0, 0, recipient, part_payload_cs, need_fwd);
}
```

Suggestion:

It is recommended to check the `adjusted_amount` before deducting the `fee`.

Resolution:

The client adopted the suggestion and fixed this issue.

POO-6 Lack of Native Token Swap Limit Check

Severity: Medium

Status: Fixed

Code Location:

contracts/swap/pool/pool.fc#137

Descriptions:

When calling this function in the native token pool, there is no check on the number of tokens, which is required for jetton type tokens.

```
;; check if the swap amount exceeds the limit  
throw_if(error::swap_limit_exceed, transfer_amount > swap_limit);
```

Suggestion:

It is recommended to confirm if it aligns with the design.

Resolution:

The client adopted the suggestion and fixed this issue.

POO-7 Incorrect Exception Throwing

Severity: Minor

Status: Fixed

Code Location:

contracts/swap/pool/pool.fc#856,895

Descriptions:

In the process of performing integer operations:

```
if (is_native) {  
    ton_amount = (msg_value - SEND_MESSAGE_TO_MESSENGER_FEE_CONSUMPTION -  
transfer_amount);  
    mode = SEND_MODE_REGULAR;  
    ;; ton native  
    throw_if(error::cross_amount_exceeded_deposit, transfer_amount > msg_value);  
}
```

There are two issues in the above code. The first is to put the throw after the operation, which will make the throw statement invalid because the exception has been thrown by the virtual machine before the exception is thrown. The second is that

`SEND_MESSAGE_TO_MESSENGER_FEE_CONSUMPTION` is not used as a size judgment operation, which may cause an exception to not be thrown in some cases.

Suggestion:

It is recommended to move the throw statement forward and include constants in size comparisons.

Resolution:

The client adopted the suggestion and fixed this issue.

POO-8 Error Code Not Used

Severity: Minor

Status: Fixed

Code Location:

contracts/swap/pool/pool.fc#8,11,20,23,27,28,31;

contracts/messenger.fc#9

Descriptions:

These error codes are never used. In the `pool::call_back_from_swap_consumer`, error code `swap_consumer_already_registered` is misused as `unknown_swap_consumer`, where it is a judgement on whether the message is from a consumer rather than whether it has been registered or not.

Suggestion:

It is recommended to remove these error codes as you see fit.

Resolution:

The client adopted the suggestion and fixed this issue.

POO-9 Calculating Gas Consumption without Checking for Sufficiency

Severity: Minor

Status: Fixed

Code Location:

contracts/swap/pool/pool.fc;

contracts/messenger.fc

Descriptions:

We have noticed that calculations of gas consumption almost never take into account the case of insufficient gas consumption, which can result in negative results and cause other functions to report errors, which can make it difficult to trace the problem.

Suggestion:

It is recommended to confirm if it aligns with the design.

Resolution:

The client adopted the suggestion and fixed this issue.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.

