Cyberscope

Audit Report

Ton Arena

May 2025

Repository : https://gitlab.com/bet_ton/core/-/tree/develop

Commit : 737d85e99f6b8b2e424ef3af2e10d0cf392c3b63

Audited by © cyberscope

Table of Contents

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
Critical	Highly Likely / High Impact
Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
Minor / Informative	Unlikely / Low to no Impact

Review

Repository	https://gitlab.com/bet_ton/core/-/tree/develop/
Commit	737d85e99f6b8b2e424ef3af2e10d0cf392c3b63

Audit Updates

Initial Audit	18 Mar 2025 https://github.com/cyberscope-io/audits/blob/main/tonarena/v1/audit.pdf
Corrected Phase 2	23 May 2025

Source Files

Filename	SHA256
creator.fc	e94e034f89ab63cbc4785c62b73b6a2b44b28abd5ec8693583b3fb80d107f28b
bet.fc	e94e034f89ab63cbc4785c62b73b6a2b44b28abd5ec8693583b3fb80d107f28b
jetton_bet.fc	4cd648ee2788288e5bd9a3a258aae8815a569d67c6ed76df48e3441485fd3cad
bet_utils.fc	48f6741bd766721063c8f559227162db7ae4e23d4dae4c1bc4c3df4db15b1b0b
jetton_proxy.fc	8bc70811f795b3993bf66c61a2ff8d46815737fca2c47a1aaca5f99558b2bf14
liquidity_pool.fc	3b150966d5de114292b622d6eb7f5a5cbcb5659d35c5bbcf7877d2788f13e3eb

opcodes.fc	ae957c61d477d1e1bb635b4b239c31e41a15d69652f87ecc6e65318d070db721
stages.fc	ebcd262f89896f5e7ade679953f8cc0560ff2ba3dfc62d4c1816514fb8ac188f
stdlib.fc	752f345de290e9594f3802ff51c5780c5eca2bcc955b161adaffb4be0845e31c
ticket.fc	fb2a5d98ac7ef1add84f33e904cf55c4e955e8cfab1731f65c464d58dc7f72bd

Overview

The Tonarena project has undergone an audit of its core smart contracts: `creator.fc`, `bet.fc`, `jetton_bet.fc`, `bet_utils.fc`, `jetton_proxy.fc`, `liquidity_pool.fc`, `ticket.fc`, `opcodes.fc`, `stages.fc`, `stdlib.fc`. In the following, an overview of the protocol's main functionalities is provided.

creator.fc

Local State:

`owner`, `commission_percentage`, `bet_code`, `jetton_bet_code`, `lp_address`

op :: create

When the contract receives an internal message with the `create` opcode in the `msg_body`, it extracts several pieces of information from the message. This includes a `uint256` value representing the `game_hash`, the addresses of `player1` and the `arbitrator`, a `uint64` value for the `valid_period`, relevant `bet_info`, a `uint256` for the `bet_size`, the relevant `token_Address` and `jetton_wallet`, and a boolean indicating whether an LP (liquidity pool) is used. The contract then distinguishes between the use of native tokens and a jetton. If no jetton address is specified, it calculates a bet address based on the specified `game_hash`. It sends a message to this address containing the `init` opcode, a `query_id`, a validity period, the `player1` address, the `arbitrator` address, and the `commission_percentage`. The entire message value is forwarded when sending this message. The message's value must be greater than the specified `bet_size`. On the other hand, if a jetton is specified, the contract derives the destination address using the `jetton_bet_code` and the incoming `game_hash`. It then forwards a similar message that includes the `init` opcode. In this case, the value carried by the message is transferred without any restriction on its size.

bet.fc

Local State:

`creator_address` , `game_hash` , `player1` , `player2` , `arbitrator_address` , `commission_percentage` , `valid_till` , `bet_size` , `stage` , `use_lp`

op :: init

Once the contract receives the `init` opcode, it checks that the sender of the message is the `creator_address` . If this check succeeds, it also confirms that the state variable `valid_till` is set to 0 and that the stage is set to `waiting_player2` . It then proceeds to load the validity period from the message as a `unit64` while setting the `valid_till` variable to a future instance. Information about the `player1` , the `arbitrator` address, the `commission_percentage` , the bet size, and the usage of `lp` is loaded from the message. If the latter is enabled, `player2` is set as the `lp` address. The contract then finalizes by updating its state.

op :: join

If the contract receives the `join` opcode, it requires that its state is set to `waiting_player2` to proceed and that it is within its validity period. If `lp` is enabled, it also confirms that the sender of the address is the stored address as `player2` in the global state. The contract aims to restrict access to the `lp` address with this check. If the option for `lp` is not enabled, the contract loads `player2` from the message body and requires the message to carry a value greater than `bet_size` . Finally, the contract sets the stage to `started` and updates its state.

op :: resolve

The `resolve` opcode may be received only by the `arbitrator_address` . Once this opcode is received, the contract loads `player1` 's address and a `uint` to represent `player1` 's choice from the arbitrator's message. Similarly, it loads `player2` 's address and choice from the same message. In addition, the `winner_choice` is loaded from the message. If the winner choice coincides with one of the two addresses, that address is set as the `winner_address` . The winner may receive the contract's balance.

op :: cancel

The `cancel` opcode can only be received by the arbitrator. If the contract is in the `waiting_player2` stage, it will send all its balance, excluding a commission, to `player1` and get destroyed. If it is in the `started` stage, it will refund `player1` and `player2` (excluding possible commissions) and get destroyed.

jetton_bet.fc

Local State: `creator_address` , `game_hash` , `player1` , `player2` , `arbitrator_address` , `commission_percentage` , `valid_till` , `bet_size` , `stage` , `token_address` , `jetton_wallet`

op :: init

Once the contract receives the `init` opcode, it checks that the sender of the message is the `creator_address` . If this check succeeds, it also confirms that the state variable `valid_till` is set to 0 and that the stage is set to `waiting_player1` . It then proceeds to load the validity period from the message as a `uint64` while setting the `valid_till` variable to a future instance. Information about `player1` , the `arbitrator_address` , the `commission_percentage` , the `bet_size` , the `token_address` , and the `jetton_wallet` is loaded from the message. The contract then finalizes by updating its state.

op :: join

Once the contract receives the `init` opcode, it checks that the sender of the message is the `creator_address` . If this check succeeds, it also confirms that the state variable `valid_till` is set to 0 and that the stage is set to `waiting_player1` . It then proceeds to load the validity period from the message as a `uint64` while setting the `valid_till` variable to a future instance. Information about `player1` , the `arbitrator_address` , the `commission_percentage` , the `bet_size` , the `token_address` , and the `jetton_wallet` is loaded from the message. The contract then finalizes by updating its state.

op :: resolve

The `resolve` opcode may be received only by the `arbitrator_address`. Once this opcode is received, the contract loads `player1`'s address and a `uint` to represent `player1`'s choice from the arbitrator's message. Similarly, it loads `player2`'s address and choice from the same message. In addition, the `winner_choice` is loaded from the message. If the `winner_choice` coincides with one of the two addresses, that address is set as the `winner_address`. The winner may receive the contract's balance.

op :: cancel

The `cancel` opcode can only be received by the arbitrator. If the contract is in the `waiting_player1` stage, it will send the remaining balance to the `creator_address`. If it is in the `waiting_player2` stage, it will send all its balance, excluding a commission, to `player1` and get destroyed. If it is in the `started` stage, it will refund `player1` and `player2` (excluding possible commissions) and get destroyed.

liquidity_pool.fc

Local State:

`owner`, `ticket_code`, `interest`

op :: connect

Once the contract receives the `connect` opcode, it will fail unless the message was received by the owner address. If this condition is satisfied, the contract extracts from the message's body an address as the `bet_address` and a `uint256` as the `bet_size`. It then forwards to the `bet_address` a message with opcode `join`, the `query_id`, and a TON balance equal to `bet_size`.

op :: deposit

In the case the message body includes the `deposit` opcode, the contract calculates the `ticket_address` as a function of the `ticket_code` stored in the state variables and the address of the sender of the message. The contract then forwards to that address a message including the `create_ticket` opcode, the `query_id`, the `interest`, and the value of native tokens carried with the internal message.

op :: redeem

Lastly, if the contract receives the `redeem` opcode, it loads from the message body a destination address and a `uint256` as value. Then it calculates an `expected_ticket_address` as a function of the loaded destination address and returns unless the sender of the message is that address. If all checks are confirmed, the contract sends to the destination address TON tokens equal to the loaded value with no other information.

ticket.fc

Local State: `lp` , `owner` , `value` , `start_ts`

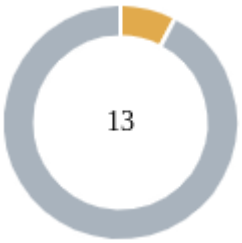
op :: create_ticket

Once the contract receives a message containing the `create_ticket` opcode, it verifies that the sender is the `lp` address. If that condition is verified, the contract loads from the incoming message a `uint7` as the interest and sets the `new_start_ts` to the current timestamp. If this is the first time this opcode is received, `start_ts` is set to the current timestamp, the storage variable named `value` is incremented by the `uint256` loaded to represent incoming coins in the `lp` from the message, and the state is saved. If this is not the first time this message is received, an `elapsed_time` is calculated from the last call, and an interest is calculated using the stored `value` , a fixed `interest_rate` , and the elapsed time. The state variable of `value` is then incremented by interest and the incoming amount, while the `start_ts` is reset to the current timestamp.

op :: redeem

This opcode may be only received by the owner of the contract. Once processed, the state variable for `value` is set to zero, and the contract sends to the `lp` address known from its state a message containing the `redeem` opcode, a `query_id` , the owner, and the zero value. The contract resets the `value` variable to 0 and self-destroys if there is no balance left.

Findings Breakdown



Critical	0
Medium	1
Minor / Informative	12

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	1	0	0
Minor / Informative	0	12	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
Medium	IMS	Inconsistent Message Structure	Acknowledged
Minor / Informative	LBF	Locked Bounced Funds	Acknowledged
Minor / Informative	CO	Code Optimization	Acknowledged
Minor / Informative	CCR	Contract Centralization Risk	Acknowledged
Minor / Informative	DRA	Duplicated Referral Addresses	Acknowledged
Minor / Informative	MAC	Missing Access Control	Acknowledged
Minor / Informative	MLVC	Missing LP Value Check	Acknowledged
Minor / Informative	MRO	Missing Refund Operation	Acknowledged
Minor / Informative	PLT	Potential Locked TON	Acknowledged
Minor / Informative	SRV	Self Reference Vulnerability	Acknowledged
Minor / Informative	TSI	Tokens Sufficiency Insurance	Acknowledged
Minor / Informative	UBD	Unchecked Bet Duration	Acknowledged
Minor / Informative	UTPD	Unverified Third Party Dependencies	Acknowledged

IMS - Inconsistent Message Structure

Criticality	Medium
Location	jetton_proxy.fc#L50
Status	Acknowledged

Description

The `jetton_proxy` contract sends a message with the `transfer()` opcode to a jetton wallet. The jetton wallet receives the message and is intended to transfer funds to the `bet_address` while also transmitting a custom payload with the `init` opcode. If the team uses a jetton wallet according to the specification provided by the TON Foundation, the transmitted message to the wallet may be inconsistent. Specifically, the jetton wallet parses the data in a different order than the one provided in the message. As a result, the transaction may fail or lead to inconsistencies in the creation of the bet.

```
cell msg = begin_cell()
.store_uint(jetton_wallet_op::transfer, 32)
.store_uint(query_id, 64)
.store_coins(jetton_amount)
.store_slice(bet_address)
.store_slice(bet_address)
.store_bool(0)
.store_coins(FORWARD_TON)
.store_ref(begin_cell().store_uint(op::join, 32).store_uint(query_id,
64).store_coins(jetton_amount).store_ref(begin_cell().store_slice(from_address
).store_uint(choice, HASH_SIZE).end_cell()).end_cell())
.end_cell();
```

Recommendation

The team is advised to review the structure of the transmitted message to ensure that it aligns with the expected message structure as specified in the jetton wallet contract provided by the TON Foundation.

https://github.com/ton-blockchain/token-contract/blob/main/ft/jetton-wallet.fc

Team Update

The team has acknowledged that this is not a security issue and states:

The finding has been fixed.

LBF - Locked Bounced Funds

Criticality	Minor / Informative
Location	creator.fc#L62
Status	Acknowledged

Description

The contract sends a bounceable message along with the associated balance. If the message bounces during the recipient's action phase, the forwarded funds return in the `creator` contract and are not returned to the external caller.

```
if (op == op::create) {
...
cell msg = begin_cell()
.store_msg_flags_and_address_none(BOUNCEABLE)
.store_slice(destination_address)
.store_coins(0)
.store_statinit_ref_and_body_ref(bet_state_init, master_msg.end_cell())
.end_cell();
send_raw_message(msg, SEND_MODE_BOUNCE_ON_ACTION_FAIL |
SEND_MODE_CARRY_ALL_REMAINING_MESSAGE_VALUE);
return ();
...
}
```

Recommendation

The contract should gracefully handle bounced messages to ensure operational consistency. The value of bounced messages should be returned to the original sender.

Team Update

The team has acknowledged that this is not a security issue and states:

Backend & Frontend assure that problem will not happen.

CO - Code Optimization

Criticality	Minor / Informative
Location	bet_utils.fc#L140
Status	Acknowledged

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations. Specifically, the contract loads the total of the hashmap although it intends to use only the first 4 entries.

```
try {
do {
(index, slice msg, int found?) = refs.udict_get_next?(REFS_KEY_SIZE,
index);
if (found?) {
slice ref = msg~load_msg_addr();
ref_addresses~tpush(ref);
total_refs += 1;
}
} until (~ found?);
} catch (_, _) {
;; skip refs if they are invalid
}
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

Team Update

The team has acknowledged that this is not a security issue and states:

Further optimizations will be made, these do not pose security issues.

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	creator#L120,127,134 liquidity_pool.fc#L55,75 ticket#L58 bet.fc#L111,149 jetton_bet.fc#L106,135
Status	Acknowledged

Description

The contract's functionality and behavior are heavily dependent on external parameters or configurations. While external configuration can offer flexibility, it also poses several centralization risks that warrant attention. Centralization risks arising from the dependence on external configuration include Single Point of Control, Vulnerability to Attacks, Operational Delays, Trust Dependencies, and Decentralization Erosion.

```
if (op == op::connect) {
...
}
```

```
if (op == op::redeem) {
...
}
```

```
if (op == op::resolve) {
...
}
if (op == op::cancel) {
...
}
```

Recommendation

To address this finding and mitigate centralization risks, it is recommended to evaluate the feasibility of migrating critical configurations and functionality into the contract's codebase itself. This approach would reduce external dependencies and enhance the contract's self-sufficiency. It is essential to carefully weigh the trade-offs between external configuration flexibility and the risks associated with centralization.

Team Update

The team has acknowledged that this is not a security issue and states:

The code is organized in a way to ease development and only constants are in common files. Regarding arbitrators - we use a centralized one for now.

DRA - Duplicated Referral Addresses

Criticality	Minor / Informative
Location	bet_utils.fc#L141
Status	Acknowledged

Description

The contract implements mechanisms to allow users to join a bet by associating with referrals. However, it is possible for a user to set the same address as a referral up to four times. This inconsistency could undermine the intended referral system of the platform, potentially leading to abuse or skewed incentives. Without checks to ensure unique referrals, the system's consistency and user trust may be compromised.

```
do {
(index, slice msg, int found?) = refs.udict_get_next?(REFS_KEY_SIZE,
index);
if (found?) {
slice ref = msg~load_msg_addr();
ref_addresses~tpush(ref);
total_refs += 1;
}
} until (~ found?);
} catch (_, _) {
;; skip refs if they are invalid
}
```

Recommendation

The team is advised to implement checks to ensure that the referrals provided by users are unique. This will enhance the consistency of the system and ensure user trust.

Team Update

The team has acknowledged that this is not a security issue and states:

Frontend & Backend handle correctness of this logic. No security issues occur.

MAC - Missing Access Control

Criticality	Minor / Informative
Location	bet.fc#L101
Status	Acknowledged

Description

The contract implements conditional segments to ensure proper code execution based on the message sender. However, even if `use_lp` is set to `false`, the contract still allows access from an LP address.

```
if (use_lp?) {
...
} else {
throw_unless(ERROR::NOT_ENOUGH_VALUE, msg_value > bet_size);
player2 = in_msg_body~load_ref();
...
}
```

Recommendation

The contract should ensure consistency of operations by implementing proper conditional checks. These checks would prevent the LP address from calling the contract when `use_lp` is set to `false`.

Team Update

The team has acknowledged that this is not a security issue and states:

This is desired logic allowing anyone to play.

MLVC - Missing LP Value Check

Criticality	Minor / Informative
Location	bet.fc#L96
Status	Acknowledged

Description

The contract implements the `join` method to allow a second player to join the bet. Additionally, it supports using an `lp_address` as a counterpart for the bet. However, in this case, the contract does not verify that the incoming message from the `lp_address` carries the necessary `msg_value` to fulfill the `bet_size`.

```
if (use_lp?) {
slice p2_parsed = player2.begin_parse();
slice lp_address = p2_parsed~load_msg_addr();
throw_unless(ERROR::NOT_FROM_LP, equal_slices_bits(sender_address,
lp_address));
} else {
...
}
```

Recommendation

The contract should verify that all incoming messages carry the necessary `msg_value` to ensure the submission of the required funds, thereby maintaining consistency in the code logic.

Team Update

The team has acknowledged that this is not a security issue and states:

This is the desired logic. The LP sends value with regards to game coefficients which are varying from time to time. It's handled by the backend's logic.

MRO - Missing Refund Operation

Criticality	Minor / Informative
Location	creator.fc#L85
Status	Acknowledged

Description

The contract initiates a new bet if the incoming message carries the necessary `bet_size` funds. If more than `bet_size` tokens are transferred, the entire message value is sent to the bet contract. However, only the `bet_size` amount can be returned.

```
if (op == op::create) {
...
if (is_address_none(token_address)) {
throw_unless(ERROR::NOT_ENOUGH_VALUE, msg_value > bet_size +
MIN_TON_FOR_BET_INITIALIZATION);
...
send_raw_message(msg, SEND_MODE_BOUNCE_ON_ACTION_FAIL |
SEND_MODE_CARRY_ALL_REMAINING_MESSAGE_VALUE);
return ();
}
```

Recommendation

The contract should manage all scenarios involving transferred funds. Specifically, it should handle cases where an excess amount is sent, ensuring that any surplus is refunded to the external user.

Team Update

The team has acknowledged that this is not a security issue and states:

The frontend handles that total value send covers only bet size + commission

PLT - Potential Locked TON

Criticality	Minor / Informative
Location	creator.fc#L114
Status	Acknowledged

Description

The contract contains TON that are unable to be transferred. Thus, it is impossible to access the locked TON. This may produce a financial loss for the users that have sent TON to the contract. Specifically, the contract allows the creation of a bet using jetton while transferring TON carried by the message to the bet address. These tokens may be inaccessible.

```
cell msg = begin_cell()
.store_msg_flags_and_address_none(BOUNCEABLE)
.store_slice(destination_address)
.store_coins(0)
.store_statinit_ref_and_body_ref(bet_state_init, master_msg.end_cell())
.end_cell();
send_raw_message(msg, SEND_MODE_BOUNCE_ON_ACTION_FAIL |
SEND_MODE_CARRY_ALL_REMAINING_MESSAGE_VALUE);
```

Recommendation

The team is advised to either remove the transfer of TON or add a withdrawal functionality. It is important to carefully consider the risks and potential issues associated with locked TON.

Team Update

The team has acknowledged that this is not a security issue and states:

The frontend handles that total value send covers only bet size + commission. If the user just sends TON to the contract then we assume he donates to the contract.

SRV - Self Reference Vulnerability

Criticality	Minor / Informative
Location	bet.fc#L117
Status	Acknowledged

Description

The contract implements mechanisms to allow users to join a bet by associating with referrals. However, it is possible for a user to set their own address as the referrer, effectively allowing them to join at a discount. This inconsistency could undermine the intended referral system of the platform.

```
if (op == op::resolve) {
...
slice winner_address = address_none();
cell refs1 = null();
if (p1.slice_bits() > 0) {
refs1 = p1~load_dict();
}
cell refs2 = null();
if (p2.slice_bits() > 0) {
refs2 = p2~load_dict();
}
...
}
```

Recommendation

Preventing self-referencing for users will ensure that all bets align with the intended design and will enhance the consistency of the system.

Team Update

The team has acknowledged that this is not a security issue and states:

The finding is managed by backend & frontend.

TSI - Tokens Sufficiency Insurance

Criticality	Minor / Informative
Location	ticket.fc#L49
Status	Acknowledged

Description

The tokens are not held within the contract itself. Instead, the contract is designed to provide the tokens from an external administrator. While external administration can provide flexibility, it introduces a dependency on the administrator's actions, which can lead to various issues and centralization risks. In particular, the contract increments its current state by an amount denoted as interest. This is a value that is not included in the contract.

In addition, the liquidity pool participates in user bets. This will lead to potential loss of funds for the users, who may not be able to redeem their deposit.

```
if (op == op::create_ticket) {
...
int interest = (holdings * interest_rate * elapsed_time) / (365 * 24 * 60 * 60 * 100);
holdings += interest;...
}
```

Recommendation

It is recommended to consider implementing a more decentralized and automated approach for handling the contract tokens. One possible solution is to hold the tokens within the contract itself. If the contract guarantees the process it can enhance its reliability, security, and participant trust, ultimately leading to a more successful and efficient process.

Team Update

The team has acknowledged that this is not a security issue and states:

Tokens cannot be held by each separate contract as interest is collected only on the liquidity pool contract.

UBD - Unchecked Bet Duration

Criticality	Minor / Informative
Location	bet.fc#L74 jetton_bet.fc#L77
Status	Acknowledged

Description

The `bet` and `jetton_bet` are initialized using user-provided parameters. These parameters include the duration of a bet, which can be set to an unrealistically large value. If such a large value is set, players may not be able to withdraw their funds unless the arbitrator invokes the cancel opcode.

```
int valid_period = in_msg_body~load_uint(64);
valid_till = valid_period + now();
```

Recommendation

It is advisable to ensure that the duration of a bet does not exceed a reasonable limit. This helps maintain consistency and ensures that users can always access their funds.

Team Update

The team has acknowledged that this is not a security issue and states:

When playing from the official frontend, bets are created with reasonable duration.

UTPD - Unverified Third Party Dependencies

Criticality	Minor / Informative
Location	creator#L66
Status	Acknowledged

Description

The contract uses an external contract in order to determine the transaction's flow. The external contract is untrusted. As a result, it may produce security issues and harm the transactions.

```
if (op == op::create) {
int game_hash = in_msg_body~load_uint(256);
cell player1 = in_msg_body~load_ref();
ensure_player_consistent(player1);
int valid_period = in_msg_body~load_uint(64);
slice bet_info = (in_msg_body~load_ref()).begin_parse();
int bet_size = bet_info~load_coins();
slice token_address = bet_info~load_msg_addr();
slice jetton_wallet = bet_info~load_msg_addr();
int use_lp? = in_msg_body~load_bool();
...}
```

Specifically, the contract relies on external information provided by the users for the token address and the jetton wallet to use. These contracts are unverified and may cause significant inconsistencies to the current execution flow.

Recommendation

The contract should use a trusted external source. A trusted source could be either a commonly recognized or an audited contract. The pointing addresses should not be able to change after the initialization.

Team Update

The team has acknowledged that this is not a security issue and states:

We allow to play with any tokens, including unverified ones.

Summary

Ton Arena contracts implement a betting mechanism. This audit investigates security issues, business logic concerns and potential improvements. Ton Arena is an interesting project that has a friendly and growing community. The team is advised to take the provided recommendations into consideration to improve the overall security and consistency of the platform.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. 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 Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

cyberscope.io