Radiance DEX smart contract analysis

Prepared by Pruvendo

08/06/21

Executive Summary

In the course of this project, we shall perform an audit of the central smart contracts (*DEXClient, DEXConnector, DEXPair, DEXRoot RootTokenContract, TONTokenWallet*) in the DEX solution developed by Radiance.

The performed audit demonstrated a high quality of the smart contract, however a number of “easy to fix” issues, including critical and major, was discovered.

Source data

The source code of the smart contracts is available on Github at:

https://github.com/radianceteam/dex2/commit/7d65f0d3b85e504ac33f01395b6ba0ffef9d5fe5

Motivation

Ideally speaking, all smart contracts issued by the Free TON ecosystem must pass the formal verification process. However, as the formal verification is an extremely time-consuming process, in some cases the business cannot wait for such a long time. Additionally, each bug found during the formal verification requires a major rework of the whole underlying stuff already completed. To address both above-mentioned issues, we suggest to undertake the informal audit before proceeding to formal verification. Being capable to find most bugs, the completion of this kind of activity allows us to:

- Release a smart contract with a high (but not the highest) level of reliability (this should be undertaken exclusively in a case of a strong business need)
- Dramatically decrease the likelihood of finding a bug during the formal verification (taking into account that each bug found at the formal verification stage brings major overhead for the proving system)

Audit outcome

Summary

The audit confirmed the high quality of the provided contract. However, a few critical and major issues were found that need to be addressed before moving forward to the next stage of formal verification as well as to limited beta production.

Most of the issues are related to either potential exceptions after `tvm.accept()` or to potential lack of gas.

All the discovered issues require just cosmetic changes of the code so no re-spin of the audit will be required before moving to the Phase 1 of formal verification.

All the issues were submitted to the development team upon the completion of internal review in a timely manner via Telegram group Radiance + ForMet but the last set of issues were reviewed just before the submission.

List of issues

All the discovered issues were grouped according to their severity and similarity. The following levels of severity were used:

- CRITICAL - the fix is absolutely mandatory as the issue can lead to the immediate losing of money or contract corruption
- MAJOR - the fix is mandatory, however can be delayed in some cases
- MINOR - the fix is optional
- NOTE - just a recommendation

№	File	Element	Severity	Description	Suggested fix
1	*DEXRoot*	*createDEXclient* *createNewPair*	CRITICAL	require after accept. In case of triggering can lead to balance exhausting	Use accept explicitly rather than in modifier and place it after the last require
2	*DEXPair*	*tokenReceivedCallback*	CRITICAL	This condition totalSupply == 0 && balanceReserve[rootA] == 0 && balanceReserve[rootB] == 0 looks incorrect. Looks like "or" should be used instead of "and". In the current version the requires in the inline functions can trigger that will lead to balance exhausting.	Change "and" to "or"
3	*DEXPair*	*burnCallback*	CRITICAL	callbacks are removed up to three times when just one added. Eventually it will lead to the full exhausting of the callback list	Remove extra callback removals
4	*TONTokenWalletConstants*	*target_gas_balance*	MAJOR	So small value looks dangerous in case gas price increases in future.	It's recommended to increase it to 1 TON
	DEXConnector *DEXClient* *DEXPair*	*constants*	MAJOR	The constants are too low to be safe	Multiply them by three
5	*TONTokenWallet*	*deployEmptyWallet*	MAJOR	In case of no accept external messages do not work	Insert tvm.accept() as well as a require to check if the balance is sufficient

6	*DEXRoot*	*createDEXclient*	MAJOR	If DEXClient does not start due to any reason (lack of gas or so) it still added to all the mappings and marked as created	Introduce a special callback that is invoked upon successful setup of DEXClient
		createDEXPair		If DEXPair does not start due to any reason (lack of gas or so) it still added to all the mappings and marked as created	Introduce a special callback that is invoked upon successful setup of DEXPair
7	*DEXPair*	*inline functions*	MAJOR	Require in inlines after accept. While normally they never should trigger in case they do it will lead to balance exhausting	Make all the assertions before accepting gas

		sendExpectedWalletAddress			
		deployWallet			
		deployEmptyWallet			
		proxyBurn		No check if the msg.value is big enough. The risk of abnormal abruption of the message chain. It's not clear if it's a real case, however, looks unsafe	Introduce require to check msg.value
	RootTokenContract	*mint*		What happens if accept message is not properly handled in case of lack of gas?	Check the balance as require and provide enough balance to the message recipient (or keep it to ensure proper handling of onBounce)
		approve			
		disapprove			
		burnByOwner			
		burnByRoot			
		onBounce		No check balance is big enough to complete the transaction	Introduce require to check balance. The better idea is to always have some minimal balance
8	*TONTokenWallet*	*internalTransfer*	MINOR	No check for sufficient balance in case owner_address.value == 0	Add some additional require

		createDEXclient			
	DEXRoot	createDEXPair			
		connectPair			
		setPair			
		getConnectorAddress			
		connectRoot			
		connectCallback			
		getAllDataPreparation			
		processSwapA			
		processSwapB			
		processLiquidity			
		returnLiquidity			
		getCallback			
	DEXClient	*createNewPair*		No check balance is big enough to complete the transaction	Introduce require to check balance
9	*RootTokenContract*	*deployWallet*	MINOR	require(tokens >= 0) is always true	Replace with require(tokens > 0)
		approve			
		disapprove			
		burnByOwner			
10	*TONTokenWallet*	*burnByRoot*	MINOR	All the wallet balance is sent back to the owner. This operation looks dangerous and unnecessary	Remove such a transfer. Try to send everything but some minimal limit

11	*TONTokenWallet*	*internalTransferFrom*	MINOR	According to TonLabs 129 flag is equivalent to 128 (1 is ignored). Should be reviewed	Use flag 128 with some additional reservation
12	*DEXRoot*	*createDEXclient*	MINOR	Strange magic value 3100000 looks inappropriate	Remove this value
13	*DEXRoot*	*createDEXPair*	MINOR	0.5 TON is a dangerous limit	Increase it to at least 1 TON
14	*DEXClient*	*getFirstCallback*	MINOR	Looks like it should be an inline call	Make it inline
	RootTokenContract	*getVersion* *getDetails* *getTotalSupply* *getWalletCode*	NOTE	No need to use responsible for pure getters	Remove responsible
15	*TONTokenWallet*	*getVersion* *balance* *getDetails* *getWalletCode* *allowance*	NOTE	No need to use responsible for pure getters	Remove responsible

		createDEXclient		Requirement !(prepay < GRAMS_CREATE_DEX_CLIENT) looks poor readable	Change to prepay >= GRAMS_CREATE_DEX_CLIENT
	DEXRoot	*createDEXPair*		Requirement !(something < something) is poor readable as well as !(something==something)	Change it to something >= something
	DEXConnector	*deployEmptyWallet*		The requirement !(msg.value < GRAMS_TO_ROOT * 2) is poor readable	Change it to msg.value >= GRAMS_TO_ROOT * 2
16	*DEXPair*	*tokensReceivedCallback*	NOTE	Condition !(amountOut > balanceReserve[arg1]) is poor readable	Change it to amountOut <= balanceReserve[arg1]
	DEXRoot	*createDEXPair*			
17	*DEXClient*	*constants*	NOTE	Numbers as 500000000 are unreadable	Change it to 0.5 ton
18	*DEXRoot*	*createDEXPair*	NOTE	The finalizing transfer should be put out of if	Put it out of if
		getQuotient			
19	*DEXClient*	*getRemainder*	NOTE	The same function is used by another contract so the code is duplicated	Implement the subcontract that implements this function and inherits all the participating high-level contracts from it
		isReady			
		isReadyToProvide			
20	*DEXClient*	*createNewPair*	NOTE	Too long lines	Split them to two
21	*DEXPair*	*tokensReceivedCallback*	NOTE	Too big function	Refactor it