Wrapped Token Security Review

TokenSoft Inc

October 22, 2021

Prepared for

Will Binns
Thomas McInerney

Prepared by

Aleks Kircanski

Executive Summary

Synopsis

During the fall of 2021, TokenSoft Inc. engaged NCC Group to conduct a security assessment of the Wrapped Token smart contract. The contract is implemented in Pact, an interpreted functional language used to specify contracts on the Kadena blockchain. The reviewed smart contract implements two interfaces, the fungible and wrapped token interfaces, loosely similar to the EIP-20 and EIP-173 Ethereum standards. The contract exposes state-changing operations for transferring, minting and revoking fungible tokens, supports administrative roles, account restriction and similar actions.

The review lasted for 5 person days. One consultant worked on the project. A kickoff and readout call were held at the beginning and at the end of the project.

Scope

The scope of this review were the following files:

- `kbtc.pact`: The implementation of the Fungible V2 and Wrapped Token V1 interfaces.
- `fungible-v2.pact`: The ERC-20 token aspects.
- `wrapped-token-v1.pact`: The governance aspects of the token.

NCC Group reviewed the following commit: `f57c2b2a2daa81f8130aaa7e0fd95ac3fe4902e7` of the `wrappedfi/wrapped_token_pact` repository on Github.

Limitations

The time frame allocated for this review was somewhat limited when compared to the actual scope. The contract behavior and its resistance to MEV (Miner Extracted Value) type of attacks depend on the underlying properties of the Kadena blockchain. To short-circuit some aspects of the review, NCC Group consulted the Kadena blockchain team (and also looked at the Kadena code base).

Testing Methodology and Key Findings

It should be noted that the Pact bug landscape is unexplored and the most common bug patterns are not well known at this stage. This is due to the fact that (1) Pact is a relatively novel language and (2) many of the bug patterns common to Solidity do not apply. In addition, developers can make use of Pact's Z3 solver integration and validate contracts for known invariants.

The strategy used during the review was to search for general unexpected behavior in contract modules. Since the contract is simple, for each function, an intuition on what the expected behavior of the function was built. Such an intuition was built based on both the function code and available documentation (source code comments). The `pact` command line tool and existing unit tests were used to test some of the module behavior.

On another front, NCC Group inspected the deviations of the Wrapped Token from the EIP-20 and EIP-173 standards. Since the ERC-20 can be considered as battle-tested, these deviations can potentially be bug leads in the Pact contract.

Some of the findings uncovered during this review include:

- **Account Identifiers Tolerate All LATIN-1 Characters:** Less tech-savvy token owners may be misled into sending tokens to unintended addresses.
- **`transfer` Function Does Not Require Destination Guard:** In certain scenarios, account hijacking and stealing of user's tokens is easier than expected.
- **Denial of Service via Account Squatting:** A malicious bot on the Kadena blockchain can disrupt normal contract's functioning and, as a consequence, possibly affect the token price.

Further testing notes are included in Testing Notes on the following page.

Security review notes

In some aspects, the Fungible V2 and Wrapped-Token V1 standards are different from the related Ethereum standards, such as EIP-20 (Token standard) and EIP-173 (Contract ownership standard). In this context, a careful review of whether the Approve/Transfer attack[1] applies in any form is warranted. Some of the differences with the Ethereum-related standards include:

- **Accounts identifiers do not necessarily include public keys:** Common token implementations leverage blockchain addresses as token account addresses. This is not the case in the Wrapped Token contract, where a token address is described by a user chosen string that has to be between 3 and 255 LATIN-1 characters. In some transfer workflows, the user is also required to set the target address guard (which, in case of a key validation guard includes the public key address, similar to the account address in other blockchains). The main `transfer` function, however, does not require the sender to specify the target address guard
- **Programmable account authentication:** The ERC-20 fungible token standard does not allow programmable authentication for token transfers. The Fungible V2 standard does, as the `accounts` table stores *guard* functions, defined by the user. A *guard* can be an arbitrary construct and does not necessarily mean access to the private key material
- **Programmable role authentication:** Similar to authenticating token transfers, the Wrapped-Token V1 standard allows arbitrary guards for authenticating to roles. In Ethereum, a typical implementation of role-based contract governance would involve a `roles` map, which stores a list of addresses and their corresponding roles.

A discussion on one attempt at a Approve/Transfer style attack which appears mostly mitigated by the underlying Kadena blockchain properties can be found in finding NCC-E002845-004 on page 9.

Additional notes

In this section, we list some observations which did not make it into findings, but are still worth mentioning:

Module hash not specified in `(use module ...)` imports: As per the Pact documentation, it's possible to specify the module hash before importing it. The documentation is not very clear on what the exact purpose for specifying the hash is and likely has to do with the contract upgrade process. It may make sense to specify the hash of the used submodules once the subcontracts are fully finalized

Account restriction and revocation obstruction: Senders whose accounts are marked as restricted are prevented from transferring funds out of their account. Moreover, revocation is only possible if the account that's being "punished" is restricted. An attacker observing a transaction that will restrict their account can broadcast a withdrawal transaction before the administrator's restriction transaction materializes, thus obstructing both restriction and revocation.

Transfer revocation unimplemented for cross-chain transfers: Single-chain revocation is implemented and requires access to `RESTRICT` and `REVOKE` roles. Token users who are subject to such corrective actions can bypass them by transferring the coins to a contract on a different chain before the transaction is included in the blockchain. If the contract administrators on different chains are the same in all circumstances (i.e., if setting administrators on the same contract on different chains is guided by code which makes those two sets equal), then this is not an issue. Otherwise, cross-contract transfer may help contract users bypass corrective measures.

Unbounded token amount: Currently, the contract administrator does not have the ability to limit the amount of tokens that can be minted. Since Pact supports integers of unlimited size and with arbitrary precision, the supply is potentially unlimited. There does not appear to be any problem with this unless a specific contract use case requires limiting the amount of tokens that can be minted.

Finally, it's worth documenting that REPL-based Pact behavior can be different from the behavior on the blockchain. In particular, transaction state roll-back does not happen in Pact in case of a function call that fails. Pact however does offer a set of REPL-only features that help make testing resemble on-chain testing more closely.

[1] https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM/edit#heading=h.m9fhqynw2xvt

Table of Findings

For each finding, NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. For an explanation of NCC Group's risk rating and finding categorization, see Appendix A on page 10.

Title	ID	Risk
Account Identifiers Tolerate All LATIN-1 Characters	001	Medium
`transfer` Function Does Not Require Destination Guard	002	Medium
Denial of Service via Account Squatting	003	Medium
Potential Fraud Based on Unsuccessful Transactions	004	Informational

Finding	**Account Identifiers Tolerate All LATIN-1 Characters**
Risk	**Medium** Impact: Medium, Exploitability: Medium
Identifier	NCC-E002845-001
Category	Other
Location	`wrapped-util.pact:33`
Impact	Less tech-savvy token owners may be misled into sending tokens to unintended addresses.

Description	Traditionally, Bitcoin developers were concerned about copy-paste attacks on blockchain addresses. The concern is mostly around the fact that transfer destination addresses may not always be thoroughly validated before making transfers. Base58 encoding was used on Bitcoin addresses to remove similar-looking characters.
	
	New accounts in the Wrapped Token contract can be created explicitly via the `create-account` function, or implicitly through other methods such as `transfer-create`, `mint`, or `transfer-crosschain`. All of the account creation code paths rely on the following account name validation function:
	
	`(defun enforce-valid-account (account:string)`
	`  (enforce (> (length account) 2) "minimum account length")`
	`  (enforce (< (length account) 256) "minimum account length")`
	`  (enforce (is-charset CHARSET_LATIN1 account) "valid charset")`
	`)`
	
	The LATIN1 charset contains a number of special characters, such as the padding character, the no break here character, etc.
	
	The way these special characters are displayed by a specific application (such as the terminal, the browser or wallet software) can affect a person's ability to parse whether the supplied address is correct or not. In traditional phishing scams, domain names that look very similar to the targeted domain are created. In the case of Wrapped token addresses, addresses similar to the legitimate destination address may be used to defraud users.

Recommendation	Consider tolerating only a subset of LATIN-1 inside account identifiers, possibly leaving only alphabet characters and numbers (white space not included). In addition, consider requiring the destination address guard as a mandatory part of the destination address in all workflows.

Finding	`transfer` **Function Does Not Require Destination Guard**
Risk	**Medium** Impact: Medium, Exploitability: Medium
Identifier	NCC-E002845-002
Category	Data Validation
Location	kbtc.pact:331
Impact	In certain scenarios, account hijacking and stealing of user's tokens is easier than it should be.
Description	This finding evaluates the impact of the following contract property:

- A user may create an account using the `create-account` method, setting the account string identifier and its guard function
- Next, the same or another user may call the `transfer` function and tokens can be transferred to the account. The destination address is only the account string and not the account guard.

The question is, what an attacker attempting to insert another `create-account` transaction before the original one, can do. The goal for the attacker would be to hijack what the user believes to be their newly created account. This question should be analyzed from a bot/arbitrage perspective, as that would maximize attackers' gains.

Account creation hijacking: If the user does not ensure that their own `create-account` transaction was finalized before making the transfer, their funds may be stolen. This depends on the wallet software. The wallet software needs to validate that the original transaction that created the account is bound to the account identifier the user sees. Consider a multi-user workflow, where one user sends funds and the other user receives them, possibly cross-chain.

The address creator must bind their created address with the account creation transaction they sent and ensure that their transaction is finalized. The wallet software must be aware of short-term reorg possibility and unbind the created address if in the new blockchain branch their transaction is no longer responsible for creating the said account name. It also means that the wallet needs to have a cross-chain view into the blockchain, since in the case of cross-chain address creation, the wallet needs to ensure that the address was finalized on the destination chain.

Account hijacking in long term network splits: Clearly the network is open to traditional double-spend attacks in case of a long-term network split. The question is whether features of the Wrapped token contract make things easier for the attacker. In the traditional double-spend attack, a purchase is made on a fork of the chain and goods are recovered by the attacker. Once the network chooses the other fork as its authoritative, the transaction is voided and the attacker has to make sure that a counter-transaction is included on the other fork. This is so that the seller cannot realize the original transaction on the newly adopted fork. All in all, the traditional double-spend attack involves a fixed victim (seller).

When it comes to the Wrapped Token smart contract, an attacker with a view in both forks can track account creations in the temporary branch and hijack all the accounts on the second branch, without targeting a particular seller. Once the second branch becomes authoritative, the attacker reaps all the transfers to hijacked accounts. The main difference is that the attack does not involve a fixed seller, it can be applied to all of the account creations that happened during the time period.

Recommendation	Consider requiring the guard specification and the string account identifier as the destination address in all transfer workflows.

Finding	**Denial of Service via Account Squatting**
Risk	**Medium** Impact: Medium, Exploitability: Medium
Identifier	NCC-E002845-003
Category	Denial of Service
Location	kbtc.pact:343
Impact	A malicious bot on the Kadena blockchain can disrupt normal contract functioning which, as a consequence, may possibly affect the token price.
Description	A bot on the network can listen for all new account creations with the three functions that can create new accounts: `create-account`, `transfer-create` and `mint`. As soon as such transactions are observed in the pool, the bot can send a competing account creation transaction and possibly outperform it by setting a higher fee (depending on the underlying Kadena blockchain properties). The original transactions would then fail and the end result is that effectively no new accounts can be created inside the contract. The fact that the contract is semi-functional can impact the price of the token, which could be conceivably used to game the market and make profit. This appears particularly effective in the case of cross-chain transfers. The first step in the `transfer-crosschain` pact debits the sender. If the target account on the destination chain doesn't exist, there is plenty of time for the attacker to create it and prevent the transfer.
Recommendation	Consider enforcing a guard during account creation. The transaction that creates accounts would have to prove access to private key material for the keyset listed in the account's guard. An attacker would have a hard time duplicating account creation without access to the legitimate account private keys.

Finding	**Potential Fraud Based on Unsuccessful Transactions**
Risk	**Informational** Impact: Undetermined, Exploitability: Undetermined
Identifier	NCC-E002845-004
Category	Other
Impact	This is an informational level finding and includes a discussion on transaction TTL on Kadena. It is not fully clear whether the scenario with a fraudster seller is ruled out under all circumstances, or not.
Description	Consider the following scenario. Alice intends to buy goods from a malicious seller Eve by using wrapped tokens. • Eve provides an address and a guard to Alice • Alice's wallet checks that the address is valid and finalized in the blockchain • As soon as Alice broadcasts the payment, Eve also broadcasts a key rotation transaction which is mined before Alice's transaction goes through • Alice and Eve now observe that Alice's payment did not go through and proceed with other payment means • Eve rotates the account guard keys back to the original ones. The question is whether, under any circumstances, Eve can make use of the Alice's original payment transaction, given that it satisfies the guards Eve just reverted to. It is assumed that the only way for Alice to ensure that the transaction *did not go through* is by observing that the TTL ran out. As per discussions with the Kadena blockchain team it is expected that the Alice's original payment transaction cannot be reused by Eve, even though the Alice's original transaction failed and was not mined when it was originally broadcasted. Kadena transactions contain the creation time and TTL fields, see `ChainMeta.hs`. The creation time mitigates long-term replay, as per the Chainweb.Pact.Utils.timingsCheck function. Short-term, there is a transaction hash store which is used to recognize duplicate transactions. As such: • To avoid the long-term replay mitigation, it appears that Eve would need to resend Alice's original transaction within the `lenientTimeSlop` buffer (95 seconds), • To avoid the short-term replay mitigation (the hash store which keeps duplicates in check), time to live (TTL) needs to run out One remaining question is what happens in the unusual scenario when the transaction's TTL is extremely short. If the TTL runs out before the long-term mitigations kick in, there may still be a transaction reintroduction opportunity. Since a short TTL would be highly unusual and an indication that something is wrong on Alice's side, this finding is marked as informational. In general, transactions with short TTL in general appear risky. Suppose the TTL spans over one or two blocks only. Suppose also, once such a transaction is not mined, the sender decides that their transaction cannot be materialized and starts a different payment method (eg. on a different blockchain). Now, assuming the transaction expiry is based on the block header time, it only takes a 1-2 block reorg for the transaction to be re-applied on the Kadena blockchain, contrary to the sender's intent.
Recommendation	NCC Group suggests giving another look at this issue with the Kadena blockchain team to ensure that under no circumstances this kind of fraud with transaction replay is possible.

Appendix A: Finding Field Definitions

The following sections describe the risk rating and category assigned to issues NCC Group identified.

Risk Scale

NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. The risk rating is NCC Group's recommended prioritization for addressing findings. Every organization has a different risk sensitivity, so to some extent these recommendations are more relative than absolute guidelines.

Overall Risk

Overall risk reflects NCC Group's estimation of the risk that a finding poses to the target system or systems. It takes into account the impact of the finding, the difficulty of exploitation, and any other relevant factors.

Critical Implies an immediate, easily accessible threat of total compromise.

High Implies an immediate threat of system compromise, or an easily accessible threat of large-scale breach.

Medium A difficult to exploit threat of large-scale breach, or easy compromise of a small portion of the application.

Low Implies a relatively minor threat to the application.

Informational No immediate threat to the application. May provide suggestions for application improvement, functional issues with the application, or conditions that could later lead to an exploitable finding.

Impact

Impact reflects the effects that successful exploitation has upon the target system or systems. It takes into account potential losses of confidentiality, integrity and availability, as well as potential reputational losses.

High Attackers can read or modify all data in a system, execute arbitrary code on the system, or escalate their privileges to superuser level.

Medium Attackers can read or modify some unauthorized data on a system, deny access to that system, or gain significant internal technical information.

Low Attackers can gain small amounts of unauthorized information or slightly degrade system performance. May have a negative public perception of security.

Exploitability

Exploitability reflects the ease with which attackers may exploit a finding. It takes into account the level of access required, availability of exploitation information, requirements relating to social engineering, race conditions, brute forcing, etc, and other impediments to exploitation.

High Attackers can unilaterally exploit the finding without special permissions or significant roadblocks.

Medium Attackers would need to leverage a third party, gain non-public information, exploit a race condition, already have privileged access, or otherwise overcome moderate hurdles in order to exploit the finding.

Low Exploitation requires implausible social engineering, a difficult race condition, guessing difficult-to-guess data, or is otherwise unlikely.

Category

NCC Group categorizes findings based on the security area to which those findings belong. This can help organizations identify gaps in secure development, deployment, patching, etc.

Access Controls	Related to authorization of users, and assessment of rights.
Auditing and Logging	Related to auditing of actions, or logging of problems.
Authentication	Related to the identification of users.
Configuration	Related to security configurations of servers, devices, or software.
Cryptography	Related to mathematical protections for data.
Data Exposure	Related to unintended exposure of sensitive information.
Data Validation	Related to improper reliance on the structure or values of data.
Denial of Service	Related to causing system failure.
Error Reporting	Related to the reporting of error conditions in a secure fashion.
Patching	Related to keeping software up to date.
Session Management	Related to the identification of authenticated users.
Timing	Related to race conditions, locking, or order of operations.

Appendix B: Project Contacts

The team from NCC Group has the following primary members:

- Aleksandar Kircanski — Consultant
  aleksandar.kircanski@nccgroup.com
- Javed Samuel — Cryptography Services Practice Director
  javed.samuel@nccgroup.com

The team from TokenSoft Inc has the following primary member:

- Will Binns — TokenSoft
  will@tokensoft.io