Cyberscope

Audit Report

DEFIWAY

November 2024

Files: bridge.tact, outreq.tact, asm.tact, sign.tact, jetton.tact

Audited by © cyberscope

Table of Contents

Overview

The bridge smart contracts of DEFIWAY on the TON network, have undergone a comprehensive audit to address security vulnerabilities, ensure business logic integrity, and optimise performance, providing users with a secure and efficient experience.

The DEFIWAY bridge consists of two smart contracts: bridge.tact and outreq.tact. The outreq.tact contract is responsible for receiving external requests to withdraw amounts from the bridge. These requests have been validated by a group of external signers. The bridge.tact contract then receives these requests and verifies the validity of the signatures against known public keys. The request for withdrawals are then processed. Below, the main functionalities of both contracts are outlined:

Contract: `outreq.tact`

Actions:

Receives an incoming `Send` message and forwards an outgoing `OutInternal` message to the bridge address. Sets:

1. `OutInternal.sender = context().sender`
2. `OutInternal.out = Send.out`
3. `OutInternal.sign = Send.sign`

Incoming messages:

```
message Send {
 out: Out;
 sign: Sign;
}

struct Out {
 queryId: Int as uint64;
 vault: Address;
 recipient: Address;
 amount: Int as coins;
 gas: Int as coins;
}

struct Sign {
 deadline: Int;
 signatures: map<Int as uint8, Cell>;
}
```

Outgoing messages

```
message OutInternal {
 out: Out;
 sign: Sign;
 sender: Address;
}
```

Contract: `bridge.tact`

Actions

Receives `Pay` messages and withholds `Pay.amount`, any excess is returned to the sender.

Receives `OutInternal` messages from the `outreq` contract.

1. Verifies that the address of the sender is the same as that of the `outreq` contract derived for the provided `OutInternal.out.queryId`.
2. Verifies `OutInternal.out.Sender` is an approved `sender`.
3. Verifies that the `OutInternal.sign` includes valid signatures from all approved signers.
4. If `OutInternal.out.vault` is not the zero address, the contract sends `OutInternal.out.gas` to the `vault` along with the remaining balance of the incoming message.
5. If `OutInternal.out.vault` is the zero address, the contract sends to `OutInternal.out.recipient`, tokens equal to `OutInternal.out.amount`. Then sends the remaining balance of the incoming message to `OutInternal.out.sender`. This is the source of the `Send` message received from the `outreq` contract.

Receives `SetSenders` messages.
1. Verifies the list of new senders is signed by all current signers.
2. Sets the new senders as the approved senders recognised by the contract.

Receives `SetSigners` messages.
1. Verifies the list of new signers is signed by all current signers.
2. Sets the new signers as the approved signers recognised by the contract.

Incoming messages

```
message OutInternal {...}
message TokenNotification {...}
message Pay {
 uuid: Int as uint128;
 amount: Int as coins;}
message SetSenders {
 new_senders: map<Address, Bool>;
 setterSeqno: Int;
 sign: Sign;}
message SetSigners {
 new_signers: map<Int as uint8, Int as uint256>;
setterSeqno: Int;
 sign: Sign;}
```

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

Testing Deploy	https://testnet.tonscan.org/address/EQAxxaaPosKf2_nPrZ_RowvNbXaZgwdZFqFPFRUoilenlGQv

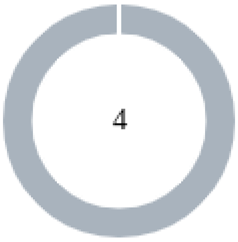
Audit Updates

Initial Audit	08 Nov 2024 https://github.com/cyberscope-io/audits/blob/main/4-defi/bridge_v1/bridge.pdf
Corrected Phase 1	1 Nov 2024

Source Files

Filename	SHA256
asm.tact	4bf9608e650d6546d8e650bc3bcd661672e86a9350e230a20a54489fab9de07e
bridge.tact	5900ad4c4837ea194098270d5cf0ce725552d0105277774db193daa7b6d7953d
jetton.tact	6eccba10c32d566a60ff1fa4fb73dbadd918d068d5625d61891ec805b5b1939e
outreq.tact	63120e41509e75207fd7fd1ad507a5c03b5263b325284e2cec77360b3d263b51
sign.tact	d6615fd05e10a7cedf1378379af4373f7601c340e0e9ae621c85e14eaf123d9e

Findings Breakdown



- Critical 0
- Medium 0
- Minor / Informative 4

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	4	0	0	0

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	CCR	Contract Centralization Risk	Unresolved
●	NBM	Non Bounceable Messages	Unresolved
●	RAR	Replay-Induced Access Restoration	Unresolved
●	UAW	Unverified Address Workchain	Unresolved

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	sign.tact#L34
Status	Unresolved

Description

The contract's functionality and behavior rely significantly on external parameters or configurations. While this external configuration provides flexibility, it introduces centralization risks that need careful consideration. These risks include a Single Point of Control, increased Vulnerability to Attacks, potential Operational Delays, Trust Dependencies, and the erosion of Decentralization. Specifically, the contracts function as a one-way bridge that does not update its state with records of user deposits. Withdrawal requests are assessed by a group of eligible signers responsible for validating these requests. These signers also have the authority to appoint new eligible signers and senders within the system. If control over the signers' private keys is compromised, it could lead to significant loss of funds.

```
fun verify(body: Cell, sender: Address, sign: Sign) {
 ...
foreach (idx, pubkey in signers) {
require(checkSignature(hash, sign.signatures.get(idx)!!.asSlice(),
pubkey),
"Invalid signature");
...
}
}
```

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.

NBM - Non Bounceable Messages

Criticality	Minor / Informative
Location	bridge.tact#L44,72,78
Status	Unresolved

Description

The bridge contract forwards funds using `bounce: false`. If an `OutInternal` message specifies the zero address for `msgOutInternal.out.vault`, the contract forwards funds to the recipient with `bounce: false`. This can lead to irretrievable loss of funds if the recipient cannot process the transfer, fails execution, or does not exist.

```
receive(msg: Pay) {
    ...
    send(SendParameters{
        to: ctx.sender,
        value: 0,
        mode: SendRemainingBalance | SendIgnoreErrors,
        bounce: false,
    });
  }
```

```
receive(msg: OutInternal) {
    ...
    send(SendParameters{
        to: msg.out.recipient,
        value: msg.out.amount,
        mode: SendPayGasSeparately | SendBounceIfActionFail,
        bounce: false,
    });
    send(SendParameters{
        to: msg.sender,
        value: 0,
        mode: SendRemainingBalance | SendBounceIfActionFail,
        bounce: false,
    });
}
```

Recommendation

Implementing bouncable messages and handling them elegantly is important. A robust solution involves receiving bounced messages and reversing the entire execution chain from the user's request in the relevant bridge contracts.

RAR - Replay-Induced Access Restoration

Criticality	Minor / Informative
Location	sign.tact#L50,65
Status	Unresolved

Description

The contract exhibits a vulnerability to replay attacks, specifically allowing replay-induced privilege reversion. This vulnerability arises from the handling of signatures associated with signer and sender accounts that hold significant privileges. The bridge contract relies on the signatures of existing signers to authorize changes to the list of approved signers and senders. However, an active signer can unilaterally revert the list of signers to a previous state by replaying an old signature. Given that a deviant signer cannot be removed from the list of active signers without the vote of an absolute majority, this vulnerability empowers such an entity to revert the system to an outdated state, potentially undermining the integrity and security of the contract's governance.

Furthermore, the contract implements a nonce mechanism within its signing process. Specifically, incoming messages include arguments for new senders or signers and a nonce. These values form the body of the message, whose validity is verified against a provided signature from the signers. However, once the state is successfully updated, the contract fails to increment the nonce value, allowing the same message to be reused at a later time.

Additionally, at the start of the signing process, the message sent by the sender is required to have a nonce larger than the contract's last known nonce. This approach could lead to inconsistencies, as the last known nonce might reach a significantly large value, thereby preventing the execution of future operations. For consistency, it is advised that the nonce increments by 1 after each successful operation and that the new nonce is exactly 1 greater than the current nonce.

```
receive(msg: SetSenders) {
  require(msg.setterSeqno > self.lastSetterSeqno, "Outdated seqno");
  self.verify(
  SetterData{ body: msg.newSenders.asCell()!!, setterSeqno: msg.setterSeqno
}.toCell(),
  context().sender,
  msg.sign
  );
  let workchain = parseStdAddress(myAddress().asSlice()).workchain;
  foreach (addr, _ in msg.newSenders) {
  require(parseStdAddress(addr.asSlice()).workchain == workchain, "Wrong
sender workchain");
  }
  self.senders = msg.newSenders;
  self.notify(emptyCell());
  }
```

```
receive(msg: SetSigners) {
  require(msg.setterSeqno > self.lastSetterSeqno, "Outdated seqno");
  self.verify(
  SetterData{ body: msg.newSigners.asCell()!!, setterSeqno: msg.setterSeqno
}.toCell(),
  context().sender,
  msg.sign
  );
  self.signers = msg.newSigners;
  self.notify(emptyCell());
  }
```

Recommendation

The team is advised to ensure that the `lastSetterSeqno` variable, which represents the nonce value, is correctly incremented with each state change. Additionally, to enhance the implementation's consistency and prevent attempts for cross-chain replication, it is recommended that the workchain ID be included as part of the signed message body.

UAW - Unverified Address Workchain

Criticality	Minor / Informative
Location	sign.tact#L65
Status	Unresolved

Description

The contract does not verify the workchain of provided addresses. The TON Blockchain consists of one masterchain and up to 2^32 workchains, each with its own rules. Currently, there are 2 workchains on TON, the MasterChain and the BaseChain. The BaseChain is used for everyday transactions between actors. Nevertheless, it is advisable to confirm the addresses are on the same workchain as the contract to ensure consistency and security.

```
receive(msg: SetSigners) {
require(msg.setterSeqno > self.lastSetterSeqno, "Outdated seqno");
self.verify(
SetterData{ body: msg.newSigners.asCell()!!, setterSeqno: msg.setterSeqno
}.toCell(),
context().sender,
msg.sign
);
self.signers = msg.newSigners;
self.notify(emptyCell());
}
```

Recommendation

The team is advised to ensure that the contract verifies the workchain of provided addresses by using the `parse_std_addr` primitive. This will help maintain transaction integrity and security. For more information, please refer to the TON documentation

Summary

The bridge contract of DEFIWAY has been audited for security vulnerabilities, business concerns and overall performance. The audit identified an issue of medium severity and several issues of minor severity affecting the overall consistency of the contract. The team is suggested to take into account these considerations to improve the security and reliability of its application.

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