Re-Review

Smart Contract Security Assessment

Vega

December 2021

Version: 1.1

Presented by:
BTblock LLC

Corporate Headquarters
BTblock LLC
2720 Xavier Street
Denver, CO 80212

Security level: Public

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

EXECUTIVE SUMMARY

Overview

Vega engaged BTBlock LLC to perform a Smart Contract Security Assessment.

The assessment was conducted remotely by the BTBlock Security Team. Testing took place on November 08 - December 03, 2021, and focused on the following objectives:

- Provide the customer with an assessment of their overall security posture and any risks discovered within the environment during the engagement.
- To provide a professional opinion on the security measures' maturity, adequacy, and efficiency.
- To identify potential issues and include improvement recommendations based on the result of our tests.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, the BTBlock Security Teams' steps to identify and validate each issue, and any applicable recommendations for remediation.

A re-review was conducted on February 18, 2022.

Key Findings

The following are the major themes and issues identified during the testing period. These, along with other items, within the findings section, should be prioritized for remediation to reduce to the risk they pose.

- BT-VEGA-01 – ERC20_Bridge_Logic_Restricted.sol - Limit on non-exempt depositors is not imposed
- BT-VEGA-02 – The Integrity of referenced addresses
- BT-VEGA-03 – MultiSigControl - Contract can reach indeterminate states
- BT-VEGA-04 – MultiSigControl - Limited number of signers
- BT-VEGA-05 – MultiSigControl – Nonces can be claimed with empty signatures
- BT-VEGA-06 – MultiSigControl – Nonce burn DOS

During the test, the following positive observations were noted regarding the scope of the engagement:

- The team was very supportive and open to discuss the design choices made

Based on formal verification we conclude that the reviewed code implements the documented functionality.

At the time of the re-review, all findings have been remediated.

Scope and Rules of Engagement

BTBlock performed a Smart Contract Security Assessment. The following table documents the targets in scope for the engagement. No other systems or resources were in scope for this assessment.

The source code was supplied through a private repository at https://gitlab.com/BTBlock-cybersec/vega with the commit hash 3f7fb2d59da27e1f7573b8095a8a59e16a0f60db. The code was audited for a second time at commit hash 4ef538239de4fc02eec70777107f818a6e98317b.

Files included in the code review

```
contracts
├── ERC20_Asset_Pool.sol*
├── ERC20_Bridge_Logic_Restricted.sol*
├── ETH_Asset_Pool.sol*
├── ETH_Bridge_Logic.sol*
├── IERC20.sol*
├── IERC20_Bridge_Logic_Restricted.sol*
├── IETH_Bridge_Logic.sol*
├── IMultisigControl.sol*
├── Migrations.sol*
└── MultisigControl.sol*
```

Table 1: Scope

TECHNICAL ANALYSES AND FINDINGS

During the Smart Contract Security Assessment, we discovered:

- 4 findings with MEDIUM severity rating.
- 1 finding with LOW severity rating.

The following chart displays the findings by severity.

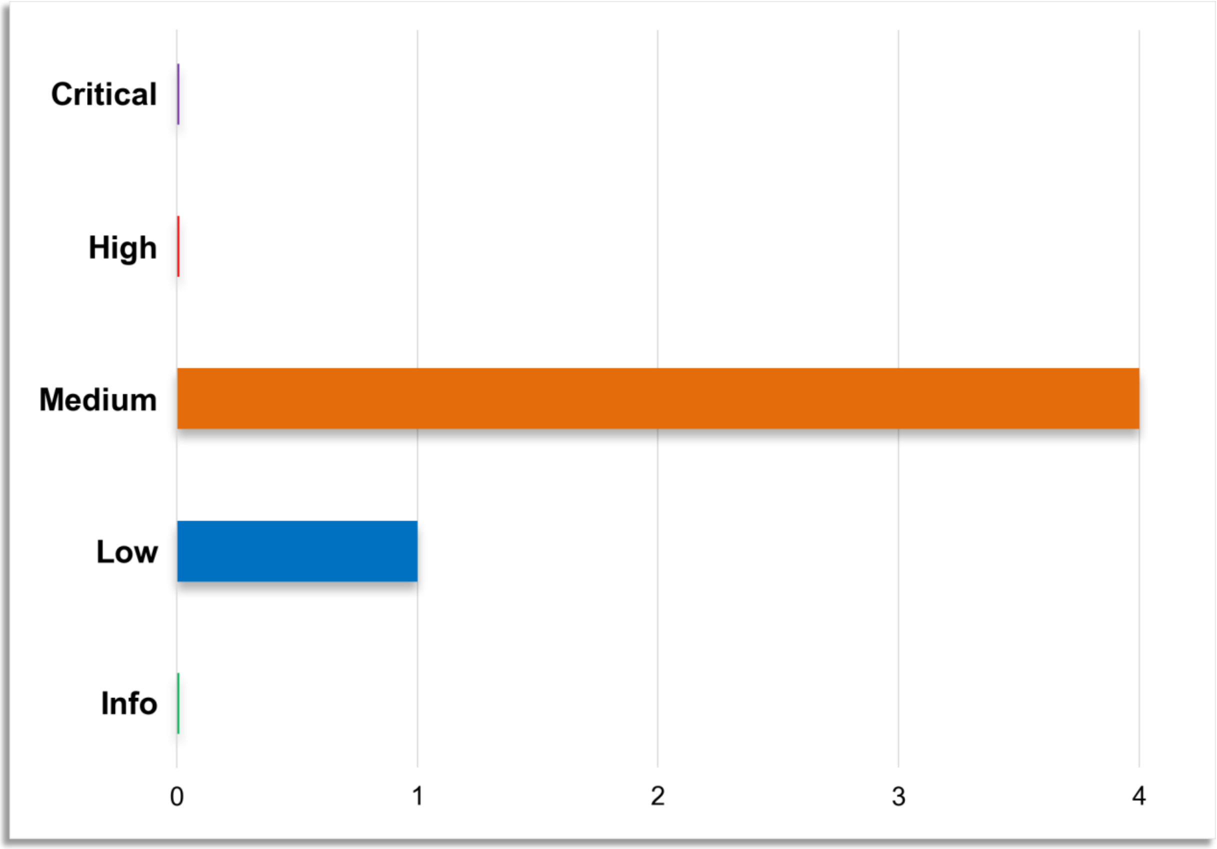

Figure 1: Findings by Severity

Findings

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

#	Severity	Description
BT-VEGA-01	**Medium**	ERC20_Bridge_Logic_Restricted.sol - Limit on non-exempt depositors is not imposed
BT-VEGA-02	**Medium**	The integrity of referenced addresses
BT-VEGA-03	**Medium**	MultiSigControl - Contract can reach indeterminate states
BT-VEGA-04	**Low**	MultiSigControl - Limited number of signers
BT-VEGA-05	**Medium**	MultiSigControl – Nonces can be claimed with empty signatures
BT-VEGA-06	**Medium**	MultisigControl - Nonce burn DOS

Table 2: Findings Overview

Technical Analyses

Based on the source code, the validity of the code was verified and confirmed that the intended functionality was implemented correctly and to the extent that the state of the repository allowed.

Based on formal verification we conclude that the code implements the documented functionality to the extent of the reviewed code.

Technical Findings

General Observations

Smart contracts provided by Vega enable Ethereum users to stake ETH and ERC20 tokens in asset pools. A signatures threshold mechanism approves access to the pools' assets and governance. The contract responsible for verifying signatures met all security requirements, missing a couple of edge cases. In general, the code was clear and very well documented.

Minor observations

The following includes a list of unstructured minor observations on the code:

- Pools' `withdraw` functions always return true, but their return value is always checked. The return value could be avoided.
- Signature control contract is cast to `MultisigControl` or `IMultisigControl` inconsistently. I would suggest using.
- MultisigControl, whose functionality is known.
- `vega_public_key` in bridge's deposit functions is only used in the event emit in without any checks.
- ERC20 bridge's `withdraw_asset` does not check whether the asset is listed.
- Some inconsistent comments in ETH_Bridge_Logic, line 16.
- The ERC20 Asset pool is not holding any assets. It simply calls IERC20.transfer in its withdrawal function.
- ETH Pool's withdrawal requirement of `address(this).balance >= amount` is redundant.
- In ERC20 Bridge, `vega_asset_ids_to_source` and `asset_source_to_vega_asset_id` are never used, only read.

ERC20_Bridge_Logic_Restricted.sol - Limit on non exempt depositors is not imposed

Finding ID: BT-VEGA-01
Severity: **Medium**
Status: **Remediated**

Description

Asset deposition is restricted to a certain amount by non-exempt users. However, the member variable checked against the deposit limits is never updated.

Proof of issue

File name: contracts/Bridge_Logic_Restricted.sol

```
// Line 103
mapping(address => mapping(address => uint256)) user_lifetime_deposits;
// Line 269
require(exempt_depositors[msg.sender] ||
user_lifetime_deposits[msg.sender][asset_source] + amount <=
asset_deposit_lifetime_limit[asset_source], "deposit over lifetime limit");
```

The above are the only uses/references of `user_lifetime_deposits`, making `user_lifetime_deposits[msg.sender][asset_source]` always 0.

Severity and Impact summary

If set, the limit on non-exempt depositors can be circumvented. It is unclear whether this could lead to unwanted loss of assets from the contracts alone.

Recommendation

Increment `user_lifetime_deposits[msg.sender][asset_source]` according to each deposited amount. Note also that `asset_deposit_lifetime_limit` could also be 0, as no check is imposed on listing.

Remediation

`user_lifetime_deposits[msg.sender][asset_source] += amount;` was added.

Integrity of referenced addresses

Finding ID: BT-VEGA-02
Severity: **Medium**
Status: **Actioned**

Description

Pools and Bridges hold addresses are referencing each other. Changes in those references can be performed only through signature verification. However, certain assumptions and/or missed checks allow space for network coordinated attacks. These are:

a. MultiSigControl addresses might differ between pools and respective bridges.
b. A pool's bridge might be addressing a different pool.
c. A bridge's pool might be addressing a different bridge.

Proof of issue

The issue is visible in `set_multisig_control`, pools' `set_bridge_address`, and bridge's constructor, where no checks are performed on the assigned addressed.

Severity and Impact summary

Missing checks on references could lead to vulnerable and/or unusable contracts

Recommendation

Assert all required checks.

Action

Part a. was remediated: Bridges' multisig_control_address has now been turned into a view function that returns the address of the owned by the pool. Parts b. and .c remain unchecked.

MultiSigControl - Contract can reach indeterminate states

Finding ID: BT-VEGA-03
Severity: **Medium**
Status: **Actioned**

Description

There are two ways the contract can reach a point of no return: a. If all signers (including the last one) remove themselves b. Suppose the threshold is set to require unanimous voting (i.e., the threshold is set to 100%).

Regarding the former, it is considered by Vega as desired functionality. We mention it in the report for the issue's completeness. The latter point is caused by "greater than" rather than a "greater or equals than" comparison.

Proof of issue

File name: contracts/MultisigControl.sol

```
// Line 62 (a)
    function remove_signer(address old_signer, uint256 nonce, bytes
calldata signatures) public override {
        ...
        signer_count--;
// Line 33 (b)
function set_threshold(uint16 new_threshold, uint256 nonce, bytes
calldata signatures) public override{
        require(new_threshold <= 1000 && new_threshold > 0, "new
threshold outside range");
// Line 123 (b)
        return ((uint256(sig_count) * 1000) / (uint256(signer_count))) >
threshold;
```

In regards to (b) if the threshold is set to 1000, verification will never be achieved

Severity and Impact summary

The repercussions of (possibly unwillingly) reaching a blocked state affect the functionality of all contracts using it.

Recommendation

a. Prevent the contract from removing its last signer.
b. Use >= to compare signature count against the threshold.

Action

Part a. was not addressed as it is considered part of the contract's functionality. Part b. was addressed by checking the threshold during `set_threshold` to be < 1000.

MultiSigControl - Limited number of signers

Finding ID: BT-VEGA-04
Severity: **Low**
Status: **Rejected**

Description

MultiSigControl is a contract responsible for verifying signatures for messages originating from multiple functions. Verifying a message checks whether signatures are accumulated exceed a threshold. The counter used to count valid signatures is **uint8** and therefore limits the maximum number of signers to 255, as overflow will revert transactions.

According to Vega's team, the signers are expected to stay well below that number in practice, but the issue is still listed as a low finding.

Proof of issue

File name: contracts/MultisigControl.sol
Line number: 86

```
        uint8 sig_count = 0;
...
            if(signers[recovered_address] &&
!has_signed[message_hash][recovered_address]){
                has_signed[message_hash][recovered_address] = true;
                sig_count++;
            }
```

`sig_count++` will always overflow and revert transactions if it exceeds 255.

Severity and Impact summary

The number of signers is limited to 255

Recommendation

Use **uint256** for **sig_count** as it is already coasted in the final comparison. This reference was also found regarding the gas cost.

References

- uint8 vs uint256 gas cost

Rejection

In practice, the signature count will be kept much lower than 256, given the limitations of the Ethereum blockchain as well.

MultiSigControl – Nonces can be claimed with empty signatures

Finding ID: BT-VEGA-05
Severity: **Medium**
Status: **Remediated**

Description

MultiSigControl's verification function does not check whether the number of signatures to be verified is 0. Since the nonce used is recorded and not allowed to be reused, an attacker could deny either future or signed but not yet verified nonces, resulting in what could resemble a denial-of-service attack.

This does not allow for arbitrary signature verification because of the threshold control.

Proof of issue

Filename: contracts/MultisigControl.sol
Line number: 79

```
    function verify_signatures(bytes calldata signatures, bytes memory
message, uint256 nonce) public override returns(bool) {
        require(signatures.length % 65 == 0, "bad sig length");
...
        used_nonces[nonce] = true;
```

Severity and Impact summary

The number of signers is limited to 255

Recommendation

Check for signatures length to be greater than 0.

Note

This issue was not included in the draft report but was added after mentioning it to Vega's team, quickly fixing it.

Remediation

`require(signatures.length > 0, "must contain at least 1 sig");` was added.

MultiSigControl – Nonce burn DOS

Finding ID: BT-VEGA-06
Severity: **Medium**
Status: **Remediated**

Description

`verify_signatures` function processes unique nonce values. After the value is used, it is added into the storage to the `used_nonces` variable. It is possible to burn the nonce even if validation fails due to an insignificant signatures count of an operations order.

Proof of Issue

Filename: contracts/MultisigControl.sol
Line number: 122

```
used_nonces[nonce] = true;
    return ((uint256(sig_count) * 1000) / (uint256(signer_count)) >
threshold;
```

Severity and Impact Summary

If the nonce is burned without an insignificant signatures count, it would be required to gather all the signatures again with a new nonce value, which will open the possibility for double-spending.

Recommendation

It is recommended to update storage variables only after all checks are passed.