



HORANGI
CYBER SECURITY

Hashport Smart Contract Review

BCW Technologies Ltd



Document Details

Document Control

Version	Author	Change	Date of Revision
0.1	Yu Xuan Soh, Sophia Ham	Initial Document	Fri 5 Aug 2022
0.2	Jeff Ong	Document Reviewed	Thu 11 Aug 2022
1.0	Nurrashidah Tukiran	Document Released	Fri 12 Aug 2022

Distribution List

Company	Contact	Email
BCW Technologies Ltd	Tai Kersten	tai@bcw.group

Horangi Pte. Ltd. ("**Horangi**") provides this assessment report (the "**Report**") to BCW Technologies Ltd ("**BCW**") subject to the terms and conditions (the "**Terms**") at Appendix B: Terms and Conditions below.

Please read these Terms carefully.



Content

Executive Brief	4
Summary Table	5
Project Detail	6
Scope Detail	6
Project Timeline	6
Methodology	7
Horangi Smart Contract Review Methodology	7
Automated Scan	7
Manual Review	7
Reporting	8
Testing Coverage	8
Security Tools	8
Severity Ratings Classification	9
Findings Register	10
Finding Details	11
Hashport Smart Contract Review	12
Insecure Version of Solidity in Use	12
Contract Over Maximum Size Limit	14
Appendix A: Scope Detail	16
Appendix B: Terms and Conditions	19



Executive Brief

Horangi conducted a security assessment for BCW Technologies Ltd (BCW), with the goal of identifying issues that could potentially negatively impact the confidentiality, integrity or availability of BCW assets and systems.

Horangi followed an automated approach, with tools such as MythX Pro, and analysed the test results for validity. Manual checks were then performed, according to guidelines and known weaknesses against the SWC Registry (Smart Contract Weakness Classification and Test Cases). Emphasis was placed on analysing logic for workflows, to ensure checks were performed adequately where necessary, such as the validation of inputs before actions were taken, or the validation of untrusted inputs received from other contracts or client-side.

36 smart contracts were understood to be in-scope for the review. These 36 contracts were understood to be part of the Hedera <-> EVM bridge, as part of the EVM Chain, the main functionality of this would be to allow users to transfer tokens from one network to another by using a “wrapped” version of the foreign token on the native network.

It was observed that the majority of the functionalities contained within the set of smart contracts existed as standalone logic and did not reference other functionality within the set of in-scope contracts. It was not immediately clear if these would be used as standalone transactions, or were referenced by third-party smart contracts which are outside the scope of the assessment. These have been assessed as-is for any required validations as per their use cases. Due to the standalone nature of most contract files, no call graph was generated for Hashport.

The following 4 functionalities below were identified by Horangi as likely to be core functionality of the Hashport smart contracts, based on existing documentation and Hashport’s use case as a Hedera/EVM bridge:

- **lock** : lock a specific number of native tokens, specifying the receiver and target network
- **unlock** : unlock a previously locked amount of native tokens by providing an array of signatures. Signatures are verified that they are signed by the members
- **mint** : mint a specific amount of wrapped tokens by providing an array of signatures, verified that they are signed by the members
- **burn** : burn a specific amount of wrapped tokens

The 36 in-scope contracts are as follows:

- Router.sol
- WrappedERC721.sol
- WrappedERC721Pausable.sol
- WrappedToken.sol
- DiamondCutFacet.sol
- DiamondLoupeFacet.sol
- ERC721PortalFacet.sol
- FeeCalculatorFacet.sol
- GovernanceFacet.sol
- GovernanceV2Facet.sol
- OwnershipFacet.sol
- PausableFacet.sol
- PaymentFacet.sol
- RouterFacet.sol
- IDiamondCut.sol
- IDiamondLoupe.sol
- IERC173.sol
- IERC2612Permit.sol
- IERC721PortalFacet.sol
- IFeeCalculator.sol
- IGovernance.sol
- IGovernanceV2.sol
- IPausable.sol
- IPayment.sol
- IRouter.sol
- IRouterDiamond.sol
- LibDiamond.sol
- LibERC721.sol
- LibFeeCalculator.sol
- LibGovernance.sol
- LibPayment.sol
- LibRouter.sol
- AddNewFunctionFacet.sol
- ReplaceFacet.sol
- Token.sol
- TokenPausabilityFacet.sol

On a high level, it was observed that logical checks were performed, and no significant vulnerabilities were observed from both the automated and manual checks. The aforementioned standalone functionalities were noted to contain adequate validation based on Horangi's understanding of their use cases as-is, such as validation of signature arrays that were to be processed as function inputs.

Two findings were found, relating to usage of an outdated Solidity compiler which may expose the compiled contract to certain vulnerabilities, and a contract being above the size limit which may prevent its publication onto Mainnet. These minor findings were noted to be best-practice findings and do not directly and adversely impact the security posture of the Hashport smart contracts.

Summary Table

The table below lists the activities conducted during the assessment, and summarises the number of issues identified per activity, grouped according to their severity and status.

Issues Identified and Severity Ratings						
Activities	Crit	High	Med	Low	Info	Total
Hashport Smart Contract Review	0	0	0	1	1	2
Total Issues	0	0	0	1	1	2



Project Detail

Scope Detail

Detailed information on the assessment's scope can be found in the Appendix A: Scope Detail section.

Scope of Work

Assessment	In-scope Targets
Hashport Smart Contract Review	36 Smart Contracts
	Note: For more information on the in-scope smart contracts, refer to Appendix A

Project Timeline

The table below outlines the timeline for all project phases, including remediation.

Project Timeline

Date	Phase
1 - 5 August 2022	Hashport Smart Contract Review



Methodology

Horangi adopts industry best practices and methodologies, coupled with our cybersecurity expertise, to build our methodology and approach for all professional services. This section will detail out Horangi's approach and coverage for each service performed.

Horangi Smart Contract Review Methodology

Horangi consultant(s) adopt a phased testing methodology when performing Smart Contract Review engagements. The three (3) main phases are automated scan, manual review, and reporting. This approach will be performed throughout the duration of the engagement.



1. Automated Scan

This phase begins with verifying the source code project is complete and compilable. This is important for certain Static Analysis Tools to scan the source code effectively. The consultants will use open-source Static Analysis Tools to conduct the automated scan. The consultants will then review the result and remove false-positive findings. Code quality and performance findings will not be reported unless it is in the scope of the assessment.

2. Manual Review

Upon completing the first phase, the consultant will manually review the source code to identify vulnerabilities that are not usually detected by the automated Static Analysis Tools. These vulnerabilities can be business logic flaws, security control bypass or other complex vulnerabilities. As the source code can be relatively complex, the consultant will prioritise the manual review on analysing critical functions and searching for severe vulnerabilities.

3. Reporting

Upon completion of the security assessment, the consultants will draft a formal report for the relevant stakeholders to review the findings uncovered during the assessment. The report will contain assessment and vulnerability details, including issue title, risk rating, description, implication, recommendation, and evidence screenshots. A report review will be performed by a senior consultant to ensure quality assurance.

Testing Coverage

Horangi smart contract review test coverage includes the reference of well known and industry-accepted standards and frameworks such as the Smart Contract Security Standards which includes coverage of the following well known attacks against smart contracts.

- Integer Overflow and Underflow
- Reentrancy Attacks
- Silent Failing Send /Unchecked Send Attacks
- Denial of Service
- Insufficient Randomness
- Front-Running Attacks
- Time Manipulation Attacks
- Short Address Attacks
- Gas Griefing Attacks
- Business Logic Misconfigurations

Security Tools

Horangi may use any relevant tools, open-source scripts or custom tools to perform automated scanning.

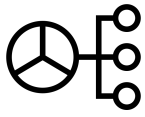
- MythX Pro
- Remix IDE
- Customised tools
- Surya

Severity Ratings Classification

Horangi uses a qualitative approach to assigning ratings to smart contract security review issues.

The impact categorization and recommended remediation timeline for the various scores is as follows:

Severity	Description
 Critical	<p>A critical severity rating is given to vulnerabilities which require immediate attention for remediation, and if compromised will significantly affect business interest.</p> <p>For critical severity vulnerabilities on production systems, Horangi recommends that remediative actions be deployed as soon as practical and not more than 15 days from the time of report. Before the issue is fully remediated, the organisation may also consider compensating controls such as black-holing the system to remove it from the network.</p>
 High	<p>A high severity rating is often given to vulnerabilities which require immediate attention for remediation, and if compromised will affect business interest.</p> <p>For high severity vulnerabilities on production systems, Horangi recommends that remediative actions be deployed within 30 days of the report.</p>
 Medium	<p>A medium severity rating is commonly given to vulnerabilities with potential to present a serious risk to business interest.</p> <p>For medium severity vulnerabilities on production systems, Horangi recommends that remediative actions be deployed within 60 days of the report.</p>
 Low	<p>A low severity rating is typically given to vulnerabilities where minimal risk to business interest is possible.</p> <p>For low severity vulnerabilities on production systems, Horangi recommends that remediative actions or compensating controls be deployed within 90 days of the report. If such actions cannot be performed within 90 days, there should be a risk assessment performed to determine if the risk is within the organisation's risk acceptance criteria.</p>
 Informational	<p>A rating of informational is typically reserved for weaknesses that represent a deviation from best practice, may expose other weaknesses or lead to future vulnerability. During the assessment period, there was no behaviour from the targets in-scope that demonstrated impact to data confidentiality, integrity or service availability.</p> <p>For informational severity vulnerabilities, Horangi recommends that vulnerabilities be remediated in a timely manner. Any remediative action should also take into account the business requirements of the organisation and the cost of remediation.</p>



Findings Register

No.	Description	Severity	Status
Smart Contract Review			
1	Insecure Version of Solidity in Use	Low	Open
2	Contract Over Maximum Size Limit	Informational	Open



Finding Details

The following section contains the details of the issues identified, grouped by the assessment activities that have been performed for this engagement.

For each assessment activity, there is a summary table of the issues discovered, sorted according to their severity. This is followed by the individual issue writeup, which contains the issue's description, the affected components, the issue's impact and appropriate remediation steps.

Hashport Smart Contract Review

Insecure Version of Solidity in Use

Severity



Low

Description

It was observed that the version of Solidity compiler in use (0.8.3) was affected by several known bugs.

The known bugs are as follows:

- Low: SOL-2022-5 - Dirty Bytes Array To Storage
- Very Low: SOL-2022-3 - Data Location Change In Internal Override
- Very Low: SOL-2022-2 - Nested Calldata Array ABIReencoding Size Validation
- Very Low: SOL-2021-3 - Signed Immutables
- Very Low: SOL-2021-2 - ABIDecode Two Dimensional Array Memory

Evidence/s

Affected Module #1

- \contracts*

A sample snippet of a contract utilising Solidity 8.3.1 has been shown below. All contracts in scope for the assessment were noted to be using Solidity compiler 8.3.1.

Code (\Router.sol - Line 2):

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.3;

import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/utils/introspection/IERC165.sol";
import "../interfaces/IDiamondLoupe.sol";
import "../interfaces/IDiamondCut.sol";
import "../interfaces/IERC173.sol";
import "../libraries/LibDiamond.sol";

contract Router {
```

[truncated]

Implication

When deploying contracts, the latest released version of Solidity should be used. Apart from exceptional cases, only the latest version of the Solidity compiler receives security fixes.


Remediation

Ensure that the latest version of Solidity compiler is used. It is also a best practice to lock pragmas to a specific compiler version throughout the contracts.

References

- <https://swcregistry.io/docs/SWC-102>
- <https://docs.soliditylang.org/en/v0.8.15/>

Contract Over Maximum Size Limit

Severity
 Informational

Description

It was observed that the compiled size of smart contract RouterFacet.sol exceeded 24576 bytes, a limit introduced in Ethereum's hard fork number 4, Spurious Dragon on 22 November 2016. As such, the contract may not be deployable on Mainnet.

Evidence/s

Affected Module #1

- \contract\facets\RouterFacet.sol

The following message shows the Remix IDE error message when RouterFacet.sol was compiled.

Screenshot:

```
Warning: Contract code size exceeds 24576 bytes (a limit introduced in Spurious
Dragon). This contract may not be deployable on mainnet. Consider enabling the
optimizer (with a low "runs" value!), turning off revert strings, or using
libraries.
--> contracts/facets/RouterFacet.sol:13:1:
|
13 | contract RouterFacet is IRouter {
| ^ (Relevant source part starts here and spans across multiple lines).
```

Implication

In this instance, it was observed that the smart contract RouterFacet.sol exceeded 24576 bytes. This limit was introduced to prevent denial-of-service (DoS) attacks. Any call to a contract is relatively cheap gas-wise, however the impact of the

contract call for Ethereum nodes increases disproportionately depending on the contract code size. Therefore, the oversized smart contract may not be deployable on Mainnet.

Remediation

Ensure that smart contracts are below the size limit of 24576 bytes. This can be done in various ways with differing impacts. More information can be found in the Reference link provided.

References

- <https://ethereum.org/en/developers/tutorials/downsizing-contracts-to-fight-the-contract-size-limit/>

Appendix A: Scope Detail

Hashport Smart Contract Review

Assessment Date(s) 1 - 5 Aug 2022

**Source Code
Package and Hash**

The following details were obtained from files in the GitHub Repository at <https://github.com/LimeChain/hashport-contracts> at 1 Aug 2022 as of commit 8125c66116d8102d801ffec21893cf6e87ee118a.

Filename: 'hashport-contracts-main.zip'

SHA256 Hash:

8739e5d3a58b4a20a3e2cb1836f4e35a33f9105ecb45041b55d14052c1c5dd18

The structure of the provided files are as follows:

```
D: .
|
| .gitignore
| hardhat.config.js
| LICENSE
| package-lock.json
| package.json
| README.md
|
|---.github
|   |---workflows
|   |   compile.yml
|   |   test.yml
|
|---contracts
|   Router.sol
|   WrappedERC721.sol
|   WrappedERC721Pausable.sol
|   WrappedToken.sol
|
|---facets
|   DiamondCutFacet.sol
|   DiamondLoupeFacet.sol
|   ERC721PortalFacet.sol
|   FeeCalculatorFacet.sol
|   GovernanceFacet.sol
|   GovernanceV2Facet.sol
|   OwnershipFacet.sol
|   PausableFacet.sol
|   PaymentFacet.sol
|   RouterFacet.sol
|
|---interfaces
|   IDiamondCut.sol
|   IDiamondLoupe.sol
|   IERC173.sol
|   IERC2612Permit.sol
|   IERC721PortalFacet.sol
|   IFeeCalculator.sol
|   IGovernance.sol
|   IGovernanceV2.sol
|   IPausable.sol
|   IPayment.sol
|   IRouter.sol
|   IRouterDiamond.sol
```



```
├── libraries
│   ├── LibDiamond.sol
│   ├── LibERC721.sol
│   ├── LibFeeCalculator.sol
│   ├── LibGovernance.sol
│   ├── LibPayment.sol
│   └── LibRouter.sol
├── mocks
│   ├── AddNewFunctionFacet.sol
│   ├── ReplaceFacet.sol
│   ├── Token.sol
│   └── TokenPausabilityFacet.sol
├── scripts
│   ├── burn-erc-20.js
│   ├── burn-erc-721.js
│   ├── deploy-router-wrapped-token.js
│   ├── deploy-router.js
│   ├── deploy-token.js
│   ├── deploy-wrapped-erc721-pausable-transfer-ownership.js
│   ├── deploy-wrapped-erc721-transfer-ownership.js
│   ├── deploy-wrapped-token.js
│   ├── erc-20-mint.js
│   ├── erc-20-unlock.js
│   ├── erc-721-mint.js
│   ├── lock-erc-20.js
│   ├── set-erc721-payment.js
│   ├── set-payment-token.js
│   ├── transfer-ownership.js
│   ├── update-member.js
│   ├── update-native-token.js
│   └── upgrade-erc721-support.js
├── test
│   └── router.js
└── util
    └── index.js
```

The 36 smart contracts in scope can be found in the contracts folder, namely:

- Router.sol
- WrappedERC721.sol
- WrappedERC721Pausable.sol
- WrappedToken.sol
- DiamondCutFacet.sol
- DiamondLoupeFacet.sol
- ERC721PortalFacet.sol
- FeeCalculatorFacet.sol
- GovernanceFacet.sol
- GovernanceV2Facet.sol
- OwnershipFacet.sol
- PausableFacet.sol
- PaymentFacet.sol
- RouterFacet.sol
- IDiamondCut.sol
- IDiamondLoupe.sol
- IERC173.sol
- IERC2612Permit.sol
- IERC721PortalFacet.sol
- IFeeCalculator.sol
- IGovernance.sol

-
- IGovernanceV2.sol
 - IPausable.sol
 - IPayment.sol
 - IRouter.sol
 - IRouterDiamond.sol
 - LibDiamond.sol
 - LibERC721.sol
 - LibFeeCalculator.sol
 - LibGovernance.sol
 - LibPayment.sol
 - LibRouter.sol
 - AddNewFunctionFacet.sol
 - ReplaceFacet.sol
 - Token.sol
 - TokenPausabilityFacet.sol

Limitations	N.A.
--------------------	------

Downtime	N.A.
-----------------	------

Appendix B: Terms and Conditions

This Report and all its contents are confidential and owned by Horangi. Provided that BCW Technologies Ltd (BCW) has fully paid all applicable fees to Horangi, Horangi grants BCW a licence to use this Report on a non-exclusive, non-sublicensable, non-transferable, worldwide, royalty-free and perpetual basis to the extent necessary for internal use by the management of BCW only. Horangi does not grant BCW the right to use any of its trademarks, trade names, or other designations.

All information in this Report is provided “as is”, without any warranties of performance, merchantability, fitness for a particular purpose, or of any other kind whether express or implied, other than those expressly stated in the Report. To the fullest extent applicable under law, Horangi disclaims all liability arising from or in connection with any decision made or action taken in reliance on the Report or its contents, and for any consequential, special, or similar damages.

Horangi Personnel have strictly confined their review to the scope and agreed hours outlined in the relevant Sales Order/Statement of Work on the relevant date(s). Horangi Personnel may identify additional observations with further time and testing, and/or with a different assessment scope. BCW acknowledges that no security assessment process, however well-planned or performed, will be free of inherent limitations and/or will be able to detect all vulnerabilities at the time it was conducted. Changes to the audited system may also result in new vulnerabilities which can only be detected by further assessments.