SOFTWARE AUDIT REPORT

for

BHOP CONSULTANTING PTE. LTD.

Prepared By: Shuxiao Wang

Hangzhou, China
December 9, 2020

Document Properties

Client	BHOP Consultanting Pte. Ltd.
Title	Software Audit Report
Target	HBTC OpenSwap
Version	0.1
Author	Ruiyi Zhang
Auditors	Ruiyi Zhang, Xuxian Jiang, Jeff Liu
Reviewed by	Jeff Liu
Approved by	Xuxian Jiang
Classification	Public

Version Info

Version	Date	Author(s)	Description
0.1	December 9, 2020	Ruiyi Zhang	Initial Release

Contact

For more information about this document and its contents, please contact PeckShield Inc.

Name	Shuxiao Wang
Phone	+86 173 6454 5338
Email	contact@peckshield.com

目录

1 介绍 4
- 1.1 关于HBTC OpenSwap . . . 4
- 1.2 关于PeckShield . . . 5
- 1.3 方法 . . . 5
- 1.4 免责声明 . . . 7

2 检测结果 9
- 2.1 总结 . . . 9
- 2.2 主要发现 . . . 10

3 检测结果详情 11
- 3.1 改进MsgAddLiquidity() 的完整性检查 . . . 11
- 3.2 可能降低价格的恶意提前交易 . . . 13
- 3.3 缺少接受流动性捐献的功能 . . . 13

4 结论 15

References 16

1 介绍

我们（**PeckShield** [7]）受客户委托对 **HBTC OpenSwap** 进行安全审计。根据我们的安全审计规范和客户需求，我们将在报告中列出用于检测潜在安全问题的系统性方法，并根据检测结果给出相应的建议或推荐以修复安全问题或提高安全性。 分析结果表明，HBTC OpenSwap 当前分支中的代码在安全性和性能上仍然有改进的空间。本文档对审计结果作了分析和阐述。

1.1 关于HBTC OpenSwap

HBTC Chain 提供了基于区块链的新一代的去中心化资产托管和清算技术。HBTC OpenSwap 模块是受到 `Uniswap`的启发， 在 HBTC Chain 中实现了一个基于 AMM 的 token 交换子系统，提供基础的DeFi服务。HBTC OpenSwap 目前是作为一个基于Cosmos的模块来实现的，通过开发一个基本的DeFi基础设施级构件，极大地推进了HBTC链的生态系统建设。

HBTC OpenSwap 的基本信息如下：

表 1.1: HBTC OpenSwap的基本信息

条目	描述
发行方	BHOP Consultanting Pte. Ltd.
官方网址	https://chain.hbtc.com/
模块	HBTC OpenSwap
合约语言	Go
审计方法	白盒
审计完成时间	December 9, 2020

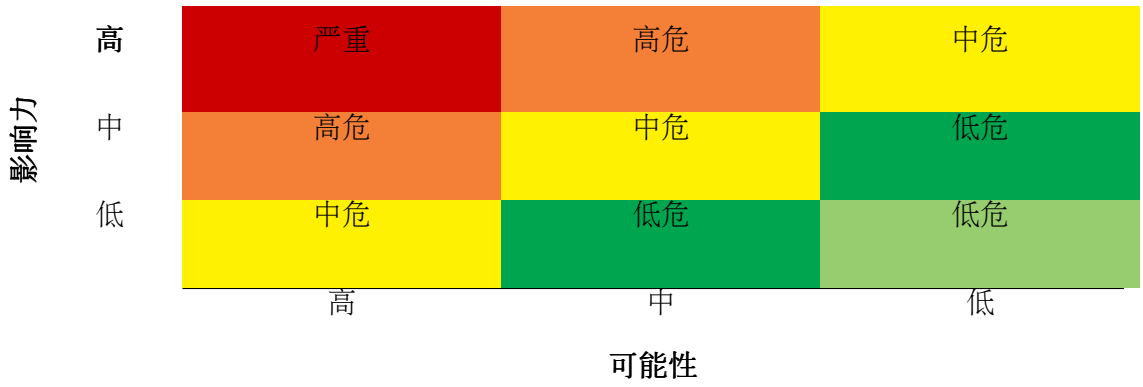
接下来，我们提供了被审计的文件的 Git 仓库链接，和被审计的分支的哈希值。

- https://github.com/hbtc-chain/bhchain/tree/main/x/openswap (74f34d0)

1.2 关于PeckShield

PeckShield (派盾) 是面向全球的业内顶尖区块链安全团队，以提升区块链生态整体的安全性、隐私性以及可用性为己任，通过发布行业趋势报告、实时监测生态安全风险，负责任曝光0day漏洞，以及提供相关的安全解决方案和服务等方式帮助社区抵御新兴的安全威胁。可以通过下列联系方式联络我们： Telegram (https://t.me/peckshield), Twitter (http://twitter.com/peckshield), or Email (contact@peckshield.com).

表 1.2: 漏洞危害性分类



1.3 方法

为了检测评估的标准化，我们根据 OWASP 风险评估方法 [6] 定义下列术语：

- 可能性 表示某个特定的漏洞被发现和利用的可能性；
- 影响力 度量了（利用该漏洞的）一次成功的攻击行动造成的损失；
- 危害性 显示该漏洞的危害的严重程度；

可能性和影响力各自被分为三个等级：高、中和低。危害性由可能性和影响力确定，分为四个等级：严重、高危、中危、低危，如表 1.2 所示。

为了评估风险，我们设置了一个检查项目清单，每个项目都会标明严重程度。对于每一个检查项目，如果我们的工具或分析没有发现任何问题，那么该合约对于该项目就被认为是安全的。对于任何被发现的问题，我们可能会进一步在我们的私有测试网中部署该合约，并运行测试来确定结果。如果有必要的话，我们会额外构造一个 PoC 来证明漏洞利用的可能性。具体的检查项目列表如表 1.3 所示。

具体来说，我们按照以下程序进行审计：

- 基本编码错误: 我们首先用我们专用的静态代码分析器静态分析给定智能合约的已知编码错误，然后手动验证（否认或确认）工具发现的所有问题。

表 1.3: 审计项目完整列表

类型	检查项目
Basic Coding Bugs	Constructor Mismatch
	Ownership Takeover
	Redundant Fallback Function
	Overflows & Underflows
	Reentrancy
	Money-Giving Bug
	Blackhole
	Unauthorized Self-Destruct
	Revert DoS
	Unchecked External Call
	Gasless Send
	Send Instead Of Transfer
	Costly Loop
	(Unsafe) Use Of Untrusted Libraries
	(Unsafe) Use Of Predictable Variables
	Transaction Ordering Dependence
	Deprecated Uses
Semantic Consistency Checks	Semantic Consistency Checks
Advanced DeFi Scrutiny	Business Logics Review
	Functionality Checks
	Authentication Management
	Access Control & Authorization
	Oracle Security
	Digital Asset Escrow
	Kill-Switch Mechanism
	Operation Trails & Event Generation
	ERC20 Idiosyncrasies Handling
	Frontend-Contract Integration
	Deployment Consistency
	Holistic Risk Management
Additional Recommendations	Avoiding Use of Variadic Byte Array
	Using Fixed Compiler Version
	Making Visibility Level Explicit
	Making Type Inference Explicit
	Adhering To Function Declaration Strictly
	Following Other Best Practices

- 语义一致性检查: 然后我们手动检查已实现的智能合约的逻辑，并与白皮书中的描述进行比较。
- 针对 DeFi 业务逻辑的专门性检查: 我们将进一步审查业务逻辑，检查系统操作，并对 DeFi 相关方面进行仔细检查，以发现可能的漏洞或错误。
- 附加建议: 我们还从经过验证的编程实践角度提供了关于智能合约编码和开发的额外建议。

为了更好地描述我们所识别的每个问题，我们将发现通过 Common Weakness Enumeration（CWE-699） [5] 进行分类，这是一个由社区开发的软件缺陷类型列表，以更好地围绕软件开发中经常遇到的概念来划分和组织缺陷。虽然 CWE-699 中使用的一些类别可能与智能合约无关，但我们使用表 1.4 中的 CWE 类别来对我们的发现进行分类。

1.4 免责声明

请注意该审计报告并不保证能够发现 HBTC OpenSwap 存在的一切安全问题，即评估结果并不能保证在未来不会发现新的安全问题。我们一向认为单次审计结果可能并不全面，因而推荐采取多个独立的审计和公开的漏洞奖赏计划相结合的方式来确保合约的安全性。最后必须要强调的是，审计结果不应构成任何投资建议。

PeckShield

表 1.4: 在本次审计中使用的 Common Weakness Enumeration (CWE) 分类

类别	总结
Configuration	Weaknesses in this category are typically introduced during the configuration of the software.
Data Processing Issues	Weaknesses in this category are typically found in functionality that processes data.
Numeric Errors	Weaknesses in this category are related to improper calculation or conversion of numbers.
Security Features	Weaknesses in this category are concerned with topics like authentication, access control, confidentiality, cryptography, and privilege management. (Software security is not security software.)
Time and State	Weaknesses in this category are related to the improper management of time and state in an environment that supports simultaneous or near-simultaneous computation by multiple systems, processes, or threads.
Error Conditions, Return Values, Status Codes	Weaknesses in this category include weaknesses that occur if a function does not generate the correct return/status code, or if the application does not handle all possible return/status codes that could be generated by a function.
Resource Management	Weaknesses in this category are related to improper management of system resources.
Behavioral Issues	Weaknesses in this category are related to unexpected behaviors from code that an application uses.
Business Logic	Weaknesses in this category identify some of the underlying problems that commonly allow attackers to manipulate the business logic of an application. Errors in business logic can be devastating to an entire application.
Initialization and Cleanup	Weaknesses in this category occur in behaviors that are used for initialization and breakdown.
Arguments and Parameters	Weaknesses in this category are related to improper use of arguments or parameters within function calls.
Expression Issues	Weaknesses in this category are related to incorrectly written expressions within code.
Coding Practices	Weaknesses in this category are related to coding practices that are deemed unsafe and increase the chances that an exploitable vulnerability will be present in the application. They may not directly introduce a vulnerability, but indicate the product has not been carefully developed or maintained.

2 检测结果

2.1 总结

在分析 HBTC OpenSwap后，我们将检测结果总结如下。如前所述，我们在第一阶段主要审计了 HBTC OpenSwap的源代码（包括相关库函数），并且使用静态分析器扫描了代码库。 这一阶段的主要目的是要识别已知的漏洞，并人工确认这些漏洞是否存在。 随后，我们会人工审查业务逻辑，检查可能存在的与业务相关的漏洞。

表 2.1: The Severity of Our Findings

严重性	发现个数	
严重	0	
高危	0	
中危	1	
低危	1	
参考	1	
总计	3	

目前为止，我们确实审计出了一些潜在的安全问题：有一些是由先前不会被考虑到的边缘情况导致的；还有一些可能会被用户非常规的交互所触发。 针对以上的每一种情况，我们都设计了相关的测试用例来复现并验证正确性。 经过了数次的分析和内部讨论之后，我们认为一些问题需要被开发人员所留意。 所有这些问题都根据它们的严重性被分类到表 2.1 中。具体细节我们将在第 3章中予以详细讨论。

2.2 主要发现

总体来讲，HBTC OpenSwap 的设计和代码实现都是优良的。但是在解决被识别出来的漏洞之前（见表 2.2），仍有一定的提升空间。在本次审计中，我们共发现1个中危漏洞、1个低危漏洞和1个参考项。

表 2.2: Key Audit Findings

编号	严重性	名称	状态
PVE-001	参考	改进MsgAddLiquidity() 的完整性检查	已确认
PVE-002	中危	可能降低价格的恶意提前交易	已确认
PVE-003	低危	缺少流动性捐献的功能	已确认

除了上述问题外，我们认为对于任何面向用户的应用和服务，建立必要的风险控制机制和制定应急预案都是非常重要的，这可能需要在主网部署前进行。风险控制机制应该在主网部署合约的那一刻开始启动。详见第 3 章。

PeckShield

3 检测结果详情

3.1 改进MsgAddLiquidity() 的完整性检查

- ID: PVE-001
- 危害性: 建议项
- 可能性: N/A
- 影响力: N/A
- 目标: `MsgAddLiquidity`
- 类型: Coding Practices [3]
- CWE 子类: CWE-391 [1]

描述

HBTC OpenSwap 模块是受到 Uniswap 的启发， 在 HBTC Chain 中实现了一个基于 AMM 的 token 交换子系统，提供基础的DeFi服务。 特别是，它定义了11种基本的消息类型，并实现了它们处理的基本逻辑。 其中，MsgAddLiquidity 和 MsgRemoveLiquidity 消息类型允许流动性提供者贡献或取消贡献流动性。

我们在下面展示了 MsgAddLiquidity 消息类型的有效性检测逻辑: ValidateBasic()。 显然，这里本应该对数值 MaxTokenAAmount 和 MaxTokenBAmount 做检查。 但代码逻辑只确保了数值 MaxTokenAAmount 是正数 (line 280)。

```
270 func (msg MsgAddLiquidity) ValidateBasic() sdk.Error {
271   if !msg.From.IsValidAddr() {
272     return sdk.ErrInvalidAddr(fmt.Sprintf("from address: %s is invalid", msg.From.String
          ()))
273   }
274   if !msg.TokenA.IsValid()  !msg.TokenB.IsValid() {
275     return sdk.ErrInvalidSymbol("invalid token symbol")
276   }
277   if msg.TokenA == msg.TokenB {
278     return sdk.ErrInvalidSymbol("token a and token b cannot be equal")
279   }
280   if !msg.MaxTokenAAmount.IsPositive()  !msg.MaxTokenAAmount.IsPositive() {
281     return sdk.ErrInvalidAmount("token amount should be positive")
282   }
283   return nil
284 }
```

Listing 3.1: openswap/types/msg.go

修复方法 修复`ValidateBasic()` 相关代码如下:

```
270 func (msg MsgAddLiquidity) ValidateBasic() sdk.Error {
271   if !msg.From.IsValidAddr() {
272     return sdk.ErrInvalidAddr(fmt.Sprintf("from address: %s is invalid", msg.From.String
          ()))
273   }
274   if !msg.TokenA.IsValid()  !msg.TokenB.IsValid() {
275     return sdk.ErrInvalidSymbol("invalid token symbol")
276   }
277   if msg.TokenA == msg.TokenB {
278     return sdk.ErrInvalidSymbol("token a and token b cannot be equal")
279   }
280   if !msg.MaxTokenAAmount.IsPositive()  !msg.MaxTokenBAmount.IsPositive() {
281     return sdk.ErrInvalidAmount("token amount should be positive")
282   }
283   return nil
284 }
```

Listing 3.2: openswap/types/msg.go

状态 该问题已被确认.

3.2 可能降低价格的恶意提前交易

- ID: PVE-002
- 危害性: 中
- 可能性: 中
- 影响力: 中
- 目标: `handler.go`
- 类型: Business Logic [4]
- CWE 子类: CWE-666 [2]

描述

需要注意的是，交换操作需满足 minAmountOut 阈值，该阈值规定了本次交易中卖出 AmountIn 的预期最小代币金额。我们指出，这样的交易可以提供一定的保护，防止价格滑落，但可能不足以抵御复杂的前期攻击，因为这些攻击可能刚好满足 minAmountOut 的要求，同时仍会导致交易用户获得较小的回报。

我们强调，这是困扰当前基于 AMM 的 DEX 方案的一个普遍问题。具体来说，一笔大额交易可能会被夹在可能会夹在两笔交易之间。其中提前的一笔卖出交易降低市场价格，后一笔交易回购以上两笔交易金额之和。不幸的是，这种夹心行为会造成亏损，给交易用户带来较小的回报，因为前面的卖出交易降低了交换利率。作为一种缓解措施，我们可以考虑对交易可能造成的滑点进行指定限制：实现一个 TWAP 或时间加权平均价格的举措（类似于 Perpetual Protocol 中的设计方法）。尽管如此，我们需要承认，该问题在很大程度上是当前区块链基础设施所固有的，仍然需要继续努力寻找有效的防御措施。

修复方法 针对上述前置攻击制定有效的缓解措施，更好地保护交易用户的利益。

状态 该问题已被确认。但为了维护主网稳定性，当前版本项目方不考虑更改实现方案。

3.3 缺少接受流动性捐献的功能

- ID: PVE-003
- 危害性: 低
- 可能性: 低
- 影响力: 低
- 目标: `liquidity`
- 类型: Business Logic [4]
- CWE 子类: CWE-666 [2]

描述

所有在 HBTC OpenSwap 中的交易对都可以以分布式、无信任的方式创建。正如在 Section 3.1 所提到，HBTC OpenSwap 提供了一个专门的消息类型 MsgCreateTradingPair 来动态创建交易对，这与最初的 Uniswap 设计是一致的。

在回顾流动性增减背后的逻辑时，我们注意到当前的执行路径不支持流动性捐赠。 虽然就目前的执行情况而言，并不存在问题，但对于是否接受流动性捐赠，确实是需要做出的设计选择。

我们的建议是，流动性捐赠有利于提高资金池估值，减少交易滑点，提供更好的用户体验。基于这一点，我们推荐项目方可以考虑支持流动性捐赠功能。

修复方法 建议支持接受流动性捐赠。

状态 该问题已被确认。但为了维护主网稳定性，当前版本项目方不考虑增加新功能。

PeckShield

4 结论

在本次安全审计中，我们分析了 HBTC OpenSwap 的设计与实现。 HBTC OpenSwap 定义实现了相关交易的处理逻辑，在 HBTC OpenSwap 链上实现了类似于 `Uniswap` 的功能。 经过进一步的分析和内部讨论，我们确定有一些问题需要被提出并给予更多的关注，这些问题在第 2 章和第 3 章中进行了详细阐述。

通过这次审计，我们认为 HBTC OpenSwap 具有良好的底层设计和工程能力。 代码库的组织逻辑清晰，模块实现优雅。我们识别出的问题都得到了及时的解决。 不论如何，HBTC OpenSwap 的软件实现得非常好，并能迅速解决审计过程中发现的问题。 除此之外，正如第 1.4 节所述，我们欢迎任何关于本报告的建设性反馈或建议。

PeckShield

References

[1] MITRE. CWE-391: Unchecked Error Condition. https://cwe.mitre.org/data/definitions/391.html.

[2] MITRE. CWE-666: Operation on Resource in Wrong Phase of Lifetime. https://cwe.mitre.org/data/definitions/666.html.

[3] MITRE. CWE CATEGORY: Bad Coding Practices. https://cwe.mitre.org/data/definitions/1006.html.

[4] MITRE. CWE CATEGORY: Business Logic Errors. https://cwe.mitre.org/data/definitions/840.html.

[5] MITRE. CWE VIEW: Development Concepts. https://cwe.mitre.org/data/definitions/699.html.

[6] OWASP. Risk Rating Methodology. https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology.

[7] PeckShield. PeckShield Inc. https://www.peckshield.com.