Six Network

SixSwap Contracts

Security Assessment

March 29th, 2021

Audited By:
Alex Papageorgiou @ CertiK
alex.papageorgiou@certik.org
Reviewed By:
Camden Smallwood @ CertiK
camden.smallwood@certik.org

Disclaimer

CertiK reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do 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.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing 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. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK'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.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	Six Network - SixSwap Contracts
Description	A cross-chain DEX implementation.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. 25d965210ed834f0c73bc70b955d073fa9a45fa4 2. 2f4edfc975ef4ced46616192d3f034f80d93329d 3. 25d965210ed834f0c73bc70b955d073fa9a45fa4

Audit Summary

Delivery Date	March 29th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	1
Timeline	March 26th, 2021 - March 29th, 2021

Vulnerability Summary

Total Issues	10
● **Total Critical**	0
● **Total Major**	2
● **Total Medium**	0
● **Total Minor**	6
● **Total Informational**	2

Executive Summary

We were tasked with auditing the codebase of the SixSwap contracts, a set of contracts enabling the cross-chain swap of the Six token between the following three networks: Stellar, Klaytn and Binance.

The codebase of the `SwapIn` suffixed implementations has a severe flaw in its design that permits anyone to arbitrarily transact funds from the contract outwards thus breaking the functionality of the contracts and rendering the system insecure. We advise this segment of the overall design to be further evaluated and potentially refactored.

Additionally, certain inconsistencies were observed as well as inapplicacies of best security practices that we pointed out and we advise the SixSwap team to assimilate in the codebase. We should note that the Stellar implementation of the swap contracts was not in scope and does not exist within the repository.

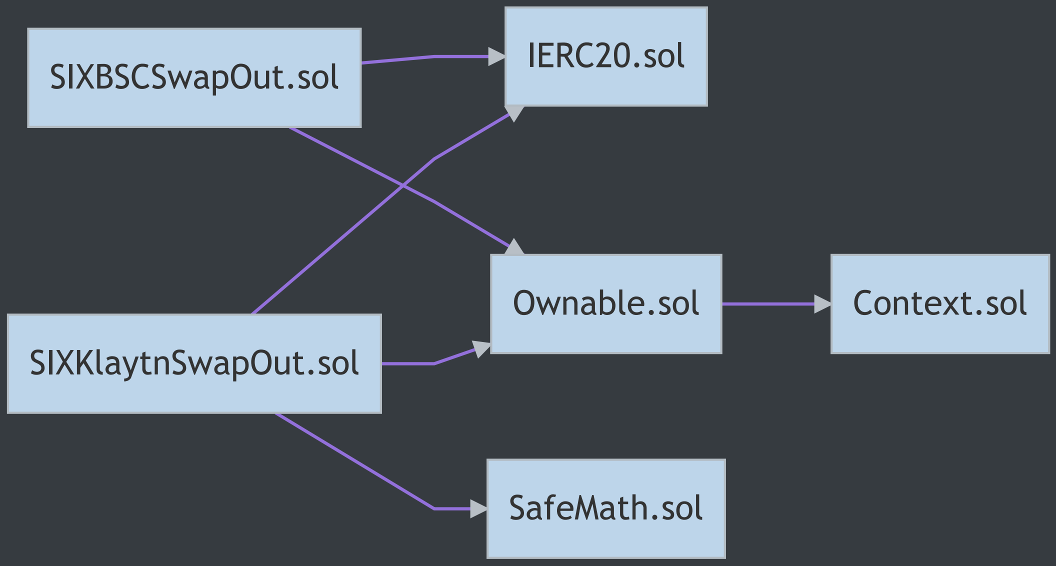
System Analysis

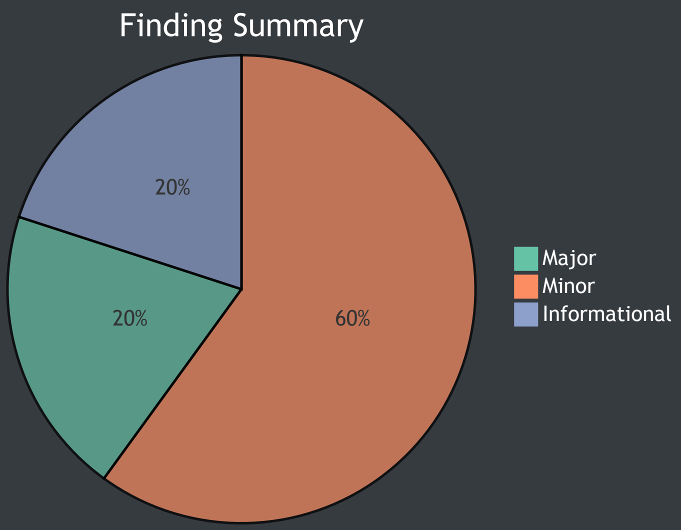
The creator of the `SwapIn` contracts can arbitrarily transfer funds from the contracts at will, persumably as a fail-safe scenario, in addition to being able to adjust the limit per transaction. The owner of the `SwapOut` suffixed contracts is able to set the wallet address, amount limit per transfer, fee transfer and transaction fee per destination chain at will to manage the system's overall operation.

Files In Scope

ID	Contract	Location
SIX	SIXBSCSwapIn.sol	bsc/contracts/SIXBSCSwapIn.sol
SIB	SIXBSCSwapOut.sol	bsc/contracts/SIXBSCSwapOut.sol
SIK	SIXKlaytnSwapIn.sol	klaytn/contracts/SIXKlaytnSwapIn.sol
SIS	SIXKlaytnSwapOut.sol	klaytn/contracts/SIXKlaytnSwapOut.sol
CON	Context.sol	bsc/contracts/utils/Context.sol
OWN	Ownable.sol	bsc/contracts/utils/Ownable.sol
OWA	Ownable.sol	klaytn/contracts/utils/Ownable.sol
SMH	SafeMath.sol	klaytn/contracts/utils/SafeMath.sol

File Dependency Graph




Finding Summary
20%
20%
60%
Major
Minor
Informational

Manual Review Findings

ID	Title	Type	Severity	Resolved
SIX-01	Incorrect Implementation	Logical Issue	🟠 Major	✓
SIX-02	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Volatile Code	🔵 Minor	◔
SIB-01	Potentially Malfunctioning Implementation	Logical Issue	🔵 Minor	◔
SIB-02	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Volatile Code	🔵 Minor	◔
SIB-03	Redundant Fee Setting	Gas Optimization	🟢 Informational	◔
SIK-01	Incorrect Implementation	Logical Issue	🟠 Major	✓
SIK-02	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Volatile Code	🔵 Minor	◔
SIS-01	Potentially Malfunctioning Implementation	Logical Issue	🔵 Minor	◔
SIS-02	Unchecked Value of ERC-20 `transfer()`/`transferFrom	Volatile Code	🔵 Minor	◔

	()` Call			
SIS-03	Incorrect Comment	Inconsistency	● Informational	✓

SIX-01: Incorrect Implementation

Type	Severity	Location
Logical Issue	● Major	SIXBSCSwapIn.sol L60-L128

Description:

The `swap` function, according to the documentation diagram provided, is meant to be invoked after a `swap` function invocation on a satellite chain `SwapOut` suffixed contract. However, no access control is imposed on the function enabling anyone to transact funds at will.

Recommendation:

We advise this trait of the system to be further evaluated as the implementation at hand is unusable in a real scenario and all funds of the contract would be at risk.

Alleviation:

The `onlyOwner` modifier was properly introduced to the function at hand ensuring it conforms to its specification.

SIX-02: Unchecked Value of ERC-20 `transfer()` / `transferFrom()` Call

Type	Severity	Location
Volatile Code	● Minor	SIXBSCSwapIn.sol L115, L156

Description:

The linked `transfer()` / `transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of a proper ERC-20 implementation.

Recommendation:

As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

Alleviation:

The `transfer()` and `transferFrom()` function results of the ERC20 standard are still not validated in the codebase of Sixswap.

SIB-01: Potentially Malfunctioning Implementation

Type	Severity	Location
Logical Issue	● Minor	SIXBSCSwapOut.sol L97-L181

Description:

The `swap` implementation of the `SIXBSCSwapOut` contract acquires the outward swap fee on top of the amount transacted instead of from the amount transacted, causing an unexpected behaviour for users of the system.

Recommendation:

We advise the `_fee` to be transacted from the transferred amount to ensure users can accurately set the allowance of the contract necessary to transact.

Alleviation:

The Six Network - SixSwap Contracts development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase due to time constraints.

SIB-02: Unchecked Value of ERC-20 `transfer()` / `transferFrom()` Call

Type	Severity	Location
Volatile Code	● Minor	SIXBSCSwapOut.sol L163, L167

Description:

The linked `transfer()` / `transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of a proper ERC-20 implementation.

Recommendation:

As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

Alleviation:

The `transfer()` and `transferFrom()` function results of the ERC20 standard are still not validated in the codebase of Sixswap.

SIB-03: Redundant Fee Setting

Type	Severity	Location
Gas Optimization	● Informational	SIXBSCSwapOut.sol L63

Description:

The `constructor` of the `SIXBSCSwapOut` contract sets the fee of a BSC destination to 25 SIX redundantly so as transfers towards a BSC destination are prohibited.

Recommendation:

We advise no fee to be set in the `constructor` as setting fees to `0` is also considered redundant as that is their default value.

Alleviation:

The Six Network - SixSwap Contracts development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase due to time constraints.

SIK-01: Incorrect Implementation

Type	Severity	Location
Logical Issue	● Major	SIXKlaytnSwapIn.sol L64-L132

Description:

The `swap` function, according to the documentation diagram provided, is meant to be invoked after a `swap` function invocation on a satellite chain `SwapOut` suffixed contract. However, no access control is imposed on the function enabling anyone to transact funds at will.

Recommendation:

We advise this trait of the system to be further evaluated as the implementation at hand is unusable in a real scenario and all funds of the contract would be at risk.

Alleviation:

The `onlyOwner` modifier was properly introduced to the function at hand ensuring it conforms to its specification.

SIK-02: Unchecked Value of ERC-20 `transfer()` / `transferFrom()` Call

Type	Severity	Location
Volatile Code	● Minor	SIXKlaytnSwapIn.sol L119, L160

Description:

The linked `transfer()` / `transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of a proper ERC-20 implementation.

Recommendation:

As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

Alleviation:

The `transfer()` and `transferFrom()` function results of the ERC20 standard are still not validated in the codebase of Sixswap.

SIS-01: Potentially Malfunctioning Implementation

Type	Severity	Location
Logical Issue	● Minor	SIXKlaytnSwapOut.sol L109-L185

Description:

The `swap` implementation of the `SIXKlaytnSwapOut` contract acquires the outward swap fee on top of the amount transacted instead of from the amount transacted, causing an unexpected behaviour for users of the system.

Recommendation:

We advise the `_fee` to be transacted from the transferred amount to ensure users can accurately set the allowance of the contract necessary to transact.

Alleviation:

The Six Network - SixSwap Contracts development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase due to time constraints.

SIS-02: Unchecked Value of ERC-20 `transfer()` / `transferFrom()` Call

Type	Severity	Location
Volatile Code	🔵 Minor	SIXKlaytnSwapOut.sol L167, L171

Description:

The linked `transfer()` / `transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of a proper ERC-20 implementation.

Recommendation:

As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

Alleviation:

The `transfer()` and `transferFrom()` function results of the ERC20 standard are still not validated in the codebase of Sixswap.

SIS-03: Incorrect Comment

Type	Severity	Location
Inconsistency	● Informational	SIXKlaytnSwapOut.sol L138

Description:

The comment states that the destination chain can only be `1` or `3` however the `require` check asserts that it can only be `3` which is the Binance chain.

Recommendation:

We advise this comment to be updated to properly reflect the statements beneath it.

Alleviation:

The `require` check's statement was adjusted to properly conform to its surrounding comment.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a `constructor` assignment imposing different `require` statements on the input variables than a setter function.