ZeroEx

Settler

Security Assessment & Correctness

November 13th, 2023

Audited By:
Angelos Apostolidis
angelos.apostolidis@ourovoros.io
Camden Smallwood
camden.smallwood@ourovoros.io
Georgios Delkos
georgios.delkos@ourovoros.io

Overview

Project Summary

Project Name	0x - Settler
Website	ZeroEx
Description	A highly optimized settlement contracts utilizing `Permit2` to perform `ERC-20` token swaps
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	cfc04600bee6bcbd0393a2d7bf43cfd9ccbb7326

Audit Summary

Delivery Date	November 13th, 2023
Method of Audit	Static Analysis, Manual Review

Vulnerability Summary

● **Total Issues**	8
● **Total Major**	1
● **Total Minor**	5
● **Total Informational**	2

Executive Summary

The Settler contracts are optimized for efficient settlements using Permit2, facilitating secure and controlled token transfers. Users sign coupons to grant Settler contract the ability to move tokens, eliminating passive allowances to contracts, hence addressing some known considerations of 0x V4 and other major decentralized exchanges. Key features include:

- Utilizing Permit2 to eliminate the need for approval transactions
- Supporting traditional liquidity pool-based decentralized exchanges
- Filling OTC orders, especially those from 0x V4
- Offering distinct support for the high-volume Uniswap V2 & V3
- Enabling meta transactions and multiple actions, primarily multiple sells and one buy action

The Settler provides optimized settlement flows, including Uniswap V3 VIP, Curve, OTC orders, and meta-transactions, minimizing gas costs and custody requirements.

Notably, the Settler is designed to hold no funds after a transaction, emphasizing its flexibility and redeployability.

Our research has shown that the Settler strives for optimal asset swaps, but caution should be taken when interacting with it without ZeroEx's additional tooling, as misconfigured transactions can lead to assets being stuck in the contract due to its intricate nature.

In summary, the Settler is an efficient, secure solution for decentralized exchanges, offering essential features and optimized settlements while leveraging the robust Permit2 mechanism to enhance user security and minimize risk.

Files In Scope

Contract	Location
src/Settler.sol	https://github.com/0xProject/0x-settler/tree/cfc04600bee6bcbd0393a2d7bf43cfd9ccbb7326/src/Settler.sol
src/ISettlerActions.sol	https://github.com/0xProject/0x-settler/tree/cfc04600bee6bcbd0393a2d7bf43cfd9ccbb7326/src/ISettlerActions.sol
src/core/UniswapV2.sol	https://github.com/0xProject/0x-settler/tree/cfc04600bee6bcbd0393a2d7bf43cfd9ccbb7326/src/core/UniswapV2.sol
src/core/UniswapV3.sol	https://github.com/0xProject/0x-settler/tree/cfc04600bee6bcbd0393a2d7bf43cfd9ccbb7326/src/core/UniswapV3.sol
src/core/Permit2Payment.sol	https://github.com/0xProject/0x-settler/tree/cfc04600bee6bcbd0393a2d7bf43cfd9ccbb7326/src/core/Permit2Payment.sol
src/core/Basic.sol	https://github.com/0xProject/0x-settler/tree/cfc04600bee6bcbd0393a2d7bf43cfd9ccbb7326/src/core/Basic.sol
src/core/VIPBase.sol	https://github.com/0xProject/0x-settler/tree/cfc04600bee6bcbd0393a2d7bf43cfd9ccbb7326/src/core/VIPBase.sol
src/core/OtcOrderSettlement.sol	https://github.com/0xProject/0x-settler/tree/cfc04600bee6bcbd0393a2d7bf43cfd9ccbb7326/src/core/OtcOrderSettlement.sol
src/core/ZeroEx.sol	https://github.com/0xProject/0x-settler/tree/cfc04600bee6bcbd0393a2d7bf43cfd9ccbb7326/src/core/ZeroEx.sol
src/utils/Panic.sol	https://github.com/0xProject/0x-settler/tree/cfc0460

	0bee6bcbd0393a2d7bf43cfd9ccbb7326/src/utils/Panic.sol
src/utils/FullMath.sol	https://github.com/0xProject/0x-settler/tree/cfc04600bee6bcbd0393a2d7bf43cfd9ccbb7326/src/utils/FullMath.sol
src/utils/SafeTransferLib.sol	https://github.com/0xProject/0x-settler/tree/cfc04600bee6bcbd0393a2d7bf43cfd9ccbb7326/src/utils/SafeTransferLib.sol
src/utils/UnsafeMath.sol	https://github.com/0xProject/0x-settler/tree/cfc04600bee6bcbd0393a2d7bf43cfd9ccbb7326/src/utils/UnsafeMath.sol
src/utils/FreeMemory.sol	https://github.com/0xProject/0x-settler/tree/cfc04600bee6bcbd0393a2d7bf43cfd9ccbb7326/src/utils/FreeMemory.sol

Findings

ID	Title	Type	Severity
F-1	Incorrect memory lookup	Volatile code	major
F-2	Unused batch token transfer feature	Dead code	informational
F-3	Comprehensive code documentation	Coding style	informational
F-4	Inadequate handling of `ETH`	Logical issue	minor
F-5	Absence of a refund mechanism	Volatile code	minor
F-6	Incompatible actions and slippage control	Volatile code	minor
F-7	Settler drainage	Volatile code	minor
F-8	Arbitrary call path	Volatile code	minor

F-1: Incorrect memory lookup

Type	Severity	Location
Volatile code	major	Permit2Payment L14, L24

Description:

Both `unsafeGet` functions within the `UnsafeArray` library contain an incorrect implementation which returns invalid memory locations, leading to possible corrupt data when reading and writing to these locations.

Recommendation:

We advise retrieving the memory offset in the element lookup table following the array header.

Alleviation:

The team has opted to consider our references and fixed the issue in commit 21dc5e5.

F-2: Unused batch token transfer feature

Type	Severity	Location
Dead code	informational	Permit2Payment L98, L138, L160

Description:

The batch transfer functionality introduced in the `Permit2Payment` contract has an incomplete implementation and remains unused throughout the codebase.

Recommendation:

We advise either incorporating the aforementioned internal functions or, alternatively, considering the removal of the redundant components.

Alleviation:

The team has opted to retain the existing components in their current state, preserving the flexibility to implement the batch token transferring feature in the future.

F-3: Comprehensive code documentation

Type	Severity	Location
Coding style	informational	Generic

Description:

Given the nature and complexity of the project, it is imperative to establish comprehensive documentation. This documentation should serve as an educational resource for project users, shedding light on its features and potential pitfalls.

Recommendation:

To enhance the project's user-friendliness and security, we recommend the addition of detailed NatSpec comments throughout the codebase.

Alleviation:

The team has opted to consider our references and will add descriptive documentation upon the product's release.

F-4: Inadequate handling of `ETH`

Type	Severity	Location
Logical issue	minor	OtcOrderSettlement L207-L250

Description:

The `fillOtcOrderSelfFunded` function is incapable to accommodate `ETH` (the native asset) as a viable `buy` currency.

Recommendation:

We advise introducing specific branching logic that can properly process the native asset alongside `ERC-20` tokens, as is demonstrated in other components of the codebase. Alternatively, consideration can be given to utilizing `WETH` (Wrapped Ether) as a suitable intermediary solution.

Alleviation:

The team has acknowledged this exhibit but decided against implementing such feature, commenting on the inherent dangers of transferring native assets. They have opted to proceed with utilizing `WETH` (Wrapped Ether) as pointed out in the recommendation above.

F-5: Absence of a refund mechanism

Type	Severity	Location
Volatile code	minor	Settler L139, L257

Description:

The `execute` function of the `Settler` contract can receive `ETH` on token-based swaps (swaps where both `sell` and `buy` assets are `ERC-20` tokens), resulting in `ETH` being unintentionally trapped within the contract. Furthermore, the Settler contract operates under the assumption that no funds are retained within it at the end of a transaction execution, potentially causing such assets to be drained during the subsequent execution of any action involving a leftover asset as the `buy` token.

Recommendation:

We advise implementing a refund mechanism that will facilitate the return of any incoming `ETH` during token-based swaps.

Alleviation:

The team has acknowledged this exhibit but decided against implementing such feature, as differentiating between each `action` 's context (i.e. `sell` and `buy` assets) is not easy and gas-efficient. The team's plan is relying on the integrator of the contract to avoid transaction construction errors and considering building a refund mechanism using `TSTORE / TLOAD` to sweep any leftover tokens back to the taker once the `Dencun` hardfork lands.

F-6: Incompatible actions and slippage control

Type	Severity	Location
Volatile code	minor	Settler L139, L257

Description:

Certain actions exhibit incompatibility with the existing slippage control mechanism. This incompatibility can lead to various undesirable outcomes, including the retention of funds within the Settler, leftover fund extraction from the Settler, or even failed execution. E.g:

- The execution of an "empty" action, which should not be subject to slippage control
- Chaining multiple, yet unrelated, actions within a single execution
- Execution of valid actions with misconfigured slippage settings, resulting in the funds remaining in the Settler
- The `POSITIVE_SLIPPAGE` action, which does not support slippage control, as both are transferring the Settler's total token balance to a recipient

This exhibit stems from the inherent ability of the Settler to exclusively support one `buy` token.

Recommendation:

We advise introducing support for multiple `buy` tokens. This alteration will enable the seamless execution of unrelated actions without the aforementioned risks. Also, a potential revision of the current correlation between the `AllowedSlippage` struct and the execution process should also be considered.

Alleviation:

The team has acknowledged this exhibit but decided against implementing such feature, as chaining unrelated actions into a single call to `execute` is discouraged and at the integrator's own risk. Also, the team acknowledges that it is possible to construct calldata that sets the slippage limit incorrectly, resulting in lost funds. Additionally, omitting the slippage limit in the `slippage` parameter is explicitly suggested when performing highly-

optimized swaps through AMMs that support it (e.g. UniV2, UniV3, Curve). In those cases, slippage limit can be checked in the AMM against the amount transferred and the additional transfer incurred by `_checkSlippageAndTransfer` can be omitted. Finally, the team wishes to note that the `POSITIVE_SLIPPAGE` action only transfer excess tokens, not the whole balance of the contract.

F-7: Settler drainage

Type	Severity	Location
Volatile code	minor	Settler L139, L257

Description:

Certain actions that can drain the Settler contract from any leftover funds. E.g:

- Executing a valid `POSITIVE_SLIPPAGE` action but with a slippage control configuration set to zero
- Executing an "empty" action that lacks slippage control, with the leftover asset as the buy token
- Executing any valid action that incorporates slippage control while setting the leftover asset as the `buy` token

Recommendation:

To ensure the security of the Settler contract and protect its users, we recommend proactively educating the project's audience about any potential pitfalls when constructing Settler transactions.

Alleviation:

The team has acknowledged this exhibit and will proceed with the all of the necessary off-chain actions needed to mitigate it.

F-8: Arbitrary call path

Type	Severity	Location
Volatile code	minor	Basic L74

Description:

Executing a valid `BASIC_SELL` action exposes a path in the `basicSellToPool` function that allows arbitrary calls, which are entirely configurable by user input.

Recommendation:

We advise imposing restrictions on the types of actions that can be executed along this particular path.

Alleviation:

The team has acknowledged this exhibit but decided against implementing such feature, as this degree of flexibility is required to be able to support arbitrary liquidity sources without change to the on-chain infrastructure. Also, the settlement contract does not hold allowances nor does it hold user funds between swaps completely mitigates the risk posed by this feature. The only special permission held by the settlement contract is the ability to spend non-expired Permit2 coupons, which we prohibit from happening inside `BASIC_SELL` function.

Disclaimer

Reports made by Ourovoros are not to be considered as a recommendation or approval of any particular project or team. Security reviews made by Ourovoros for any project or team are not to be taken as a depiction of the value of the “product” or “asset” that is being reviewed.

Ourovoros reports are not to be considered as a guarantee of the bug-free nature of the technology analyzed and should not be used as an investment decision with any particular project. They 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.

Each company and individual is responsible for their own due diligence and continuous security. Our goal is to help reduce the attack parameters and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claim any guarantee of security or functionality of the technology we agree to analyze.