# 0x MultiAssetProxy Audit

# 1 Summary

ConsenSys Diligence conducted a security audit on the 0x MultiAssetProxy contract. This contract is part of 0x's decentralized exchange. This new component makes it possible to trade bundles of assets in a single trade.

## 1.1 Audit Dashboard

### Audit Details

- **Project Name:** 0x MultiAssetProxy
- **Client Name:** 0x
- **Client Contact:** Amir Bandeali
- **Auditors:** Steve Marx, John Mardlin
- **GitHub :** https://github.com/ConsenSys/0x-audit-2018-12/
- **Languages:** Solidity assembly
- **Date:** 2018-12-10

### Number of issues per severity

| | Minor | Medium | Major | Critical |
|---|---|---|---|---|
| Open | 0 | 0 | 0 | 0 |
| Closed | 1 | 1 | 0 | 0 |

## 1.2 Audit Goals

The focus of the audit was to verify that the smart contract system is secure, resilient and working according to its specifications. The audit activities can be grouped in the following three categories:

**Security:** Identifying security related issues within each contract and within the system of contracts.

**Sound Architecture:** Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.

**Code Correctness and Quality:** A full review of the contract source code. The primary areas of focus include:

- Correctness
- Readability
- Sections of code with high complexity
- Improving scalability
- Quantity and quality of test coverage

## 1.3 System Overview

### Documentation

The following documentation was available to the audit team:

- MultiAssetProxy specification
- MultiAssetProxy ZEIP

### Scope

This audit was limited in scope to just the `MultiAssetProxy` contract.

### Design

The `MultiAssetProxy` is a somewhat straightforward contract written in Solidity assembly for gas efficiency reasons. It accepts data which specifies a list of assets and corresponding amounts. It loops through those lists and transfers the requested amount of each asset by delegating to the appropriate asset proxy.

Only authorized callers may invoke the `MultiAssetProxy`, and only registered asset proxies may be delegated to.

## 1.4 Key Observations/Recommendations

- The code is clear and contains excellent comments.
- Because the `MultiAssetProxy` is backward compatible with existing asset proxies, the code changes to add this feature are minimal.
- Most of the data consumed by the `MultiAssetProxy` is not validated in the rest of the contract system. Either more validation should be added to the contract, or a

high burden for validation is being placed on the tools used by individuals to participate in trades.

# 2 Issue Overview

The following table contains all the issues discovered during the audit. The issues are ordered based on their severity. More detailed description on the levels of severity can be found in Appendix 1. The table also contains the GitHub status of any discovered issue.

| Chapter | Issue Title | Issue Status | Severity |
|---|---|---|---|
| 3.1 | Inconsistent parameter validation | Closed | Medium |
| 3.2 | Multiplication overflow check doesn't handle zero | Closed | Minor |

# 3 Issue Detail

## 3.1 Inconsistent parameter validation

| Severity | Status | Link | Remediation Comment |
|---|---|---|---|
| Medium | Closed | issues/23 | this issue was closed as WONTFIX per the 0xProject teams reasoning: |

> We decided to leave the single overflow check in for this reason: someone can create an order where fully filling the order would cause an overflow, but it would still be safe to partially fill the order. If we removed the overflow check in this case, the order would be potentially deceptive and unsafe. |

### Description

`MultiAssetProxy` accepts a complex set of parameters. These parameters are first established by a trade *maker* and then accepted by a trade *taker*. `MultiAssetProxy` offers some protection against invalid data, but this protection is inconsistent and not comprehensive.

In particular, this inconsistency makes it hard to know which component of the system is responsible for validation. Future code maintainers and users of the system may make incorrect assumptions about where validation happens.

### Examples

This line of code (and the calculations below it) could overflow if the user-provided offset is very high:

contracts/core/contracts/protocol/AssetProxy/MultiAssetProxy.sol:L112

```
let amountsOffset := calldataload(add(assetDataOffset, 40))
```

This multiplication could overflow if the provided asset array length is very high. This could be used to prevent the `for` loop from executing, resulting in nothing being transferred to the trade taker:

contracts/core/contracts/protocol/AssetProxy/MultiAssetProxy.sol:L170

```
let amountsByteLen := mul(amountsLen, 32)
```

The following code is the only overflow check in the contract:

contracts/core/contracts/protocol/AssetProxy/MultiAssetProxy.sol:L181-L185

```
let totalAmount := mul(amountsElement, amount)

// Revert if multiplication resulted in an overflow
if iszero(eq(div(totalAmount, amount), amountsElement)) {
    // Revert with `Error("UINT256_OVERFLOW")`
```

**Remediation**

Ultimately, responsibility for the effects of a trade are the responsibility of the trade maker and taker. Presumably, takers use tools to understand the trades they're making, and those tools can perform the necessary validation to reject invalid trades or clearly identify what such trades will do.

This leaves a question of how much validation should happen in the contract itself. It may be reasonable to leave the validation to off-chain tools, but for greater safety, we recommend adding more checks for the validity of data. Here are a couple examples of such validation:

1. Check for overflows when computing with user-provided values. (An easy way to do this is just check that offsets and lengths are in a reasonable range.)
2. Check that the length of provided data structures matches their specified length (e.g. that the right number of bytes have been provided to cover the `amounts` and `nestedAssetData` arrays).

If such validation is *not* to be added, the inconsistency can be resolved by removing the sole overflow check and documenting that this contract does no validation whatsoever.

### 3.2 Multiplication overflow check doesn't handle zero

| Severity | Status | Link | Remediation Comment |
| --- | --- | --- | --- |
| **Minor** | **Closed** | issues/24 | This issue was closed in pull/1455. |

#### Description

When determining the total amount of each asset to transfer, a multiplication is performed. The result of this multiplication is checked for overflow, but that check fails to handle a zero.

contracts/core/contracts/protocol/AssetProxy/MultiAssetProxy.sol:L181-L185

```
let totalAmount := mul(amountsElement, amount)

// Revert if multiplication resulted in an overflow
if iszero(eq(div(totalAmount, amount), amountsElement)) {
    // Revert with `Error("UINT256_OVERFLOW")`
```

The check will fail if `amountElement` is non-zero and `amount` is zero. (The `div` opcode returns `0` for this case, which won't match the `amountElement`.)

Note that it may be okay to reject trades that involve an `amount` of `0`, since such a trade should have no effect anyway. Trades with a `0` amount are already rejected in `MixinAssetProxyDispatcher`, but it may be good to make an explicit decision about them in this contract.

#### Remediation

Either remove this overflow check (pending a decision about parameter validation in general), explicitly reject trades with an `amount` of `0`, or update the overflow check to allow them.

## 4 Threat Model

The creation of a threat model is beneficial when building smart contract systems as it helps to understand the potential security threats, assess risk, and identify appropriate mitigation strategies. This is especially useful during the design and development of a contract system as it allows to create a more resilient design which is more difficult to change post-development.

A threat model was created during the audit process in order to analyze the attack surface of the contract system and to focus review and testing efforts on key areas that a malicious actor would likely also attack. It consists of two parts a high level design diagram that help to understand the attack surface and a list of threats that exist for the contract system.

### 4.1 Overview

Although the `MultiAssetProxy` itself does not directly own any assets, it has the ability to transfer arbitrary assets from traders to any destination by calling out to the appropriate asset proxy. This makes it a valuable target for an attacker. Robust mechanisms are in place to prevent unauthorized callers from directly invoking the contract, so any exploit would have to pass through the `Exchange` contract layer. This minimal surface area, combined with the low branching factor of the contract itself, results in a low risk of security vulnerabilities.

### 4.2 Detail

**Assets**

- All tokens authorized by traders

**Potential attackers**

- Malicious traders: maker or taker
- Malicious forwarders: third parties submitting trades
- Other actors: people directly trying to exploit the contract

**Surface area**

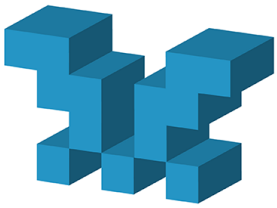
- Traders can manipulate data passed through the `Exchange` to the `MultiAssetProxy`
- Traders and relayers can manipulate other aspects of transactions (gas, stack depth, origin)
- Anyone can directly attempt calls to the `MultiAssetProxy`, including targeting the fallback function

The most promising avenues of attack are probably on the trader side. The data passed around is complex and dictates what assets are transferred where, and this data is largely controllable by traders. Makers are particularly interesting attackers because they establish the initial transaction inputs.

## 5 Tool based analysis

The issues from the tool based analysis have been reviewed and the relevant issues have been listed in chapter 3 - Issues.

### 5.1 Mythril



Mythril is a security analysis tool for Ethereum smart contracts. It uses concolic analysis to detect various types of issues. The tool was used for automated vulnerability discovery for all audited contracts and libraries. More details on Mythril's current vulnerability coverage can be found here.

The raw output of the Mythril vulnerability scan can be found here.

### 5.2 Solhint

This is an open source project for linting Solidity code. The project provides both Security and Style Guide validations. The issues of Solhint were analyzed for security relevant issues only. It is still recommended to use Solhint during development to improve code quality while writing smart contracts.

The raw output of the Solhint vulnerability scan can be found here.

### 5.3 Surya

Surya is an utility tool for smart contract systems. It provides a number of visual outputs and information about structure of smart contracts. It also supports querying the function call graph in multiple ways to aid in the manual inspection and control flow analysis of contracts.

A complete list of functions with their visibility and modifiers can be found here.

### 5.4 Odyssey

Odyssey is an audit tool that acts as the glue between developers, auditors and tools. It leverages Github as the platform for building software and aligns to the approach that quality needs to be addressed as early as possible in the development life cycle and small iterative security activities spread out through development help to produce a more secure smart contract system. In its current version Odyssey helps to better communicate audit issues to development teams and to successfully close them.

## 6 Test Coverage Measurement

Testing is implemented using the Mocha. 6244 tests are included in the test suite (for the entire contract system) and they all pass.

We were unable to generate test code coverage reports working in the provided repo. Based on a manual review of the tests, and given the low branching structure of the `MultiAssetProxy`, we believe that test coverage is likely to be very close to 100%.

# Appendix 1 - Severity

## A.1.1 - Minor

Minor issues are generally subjective in nature, or potentially deal with topics like "best practices" or "readability". Minor issues in general will not indicate an actual problem or bug in code.

The maintainers should use their own judgment as to whether addressing these issues improves the codebase.

## A.1.2 - Medium

Medium issues are generally objective in nature but do not represent actual bugs or security problems.

These issues should be addressed unless there is a clear reason not to.

## A.1.3 - Major

Major issues will be things like bugs or security vulnerabilities. These issues may not be directly exploitable, or may require a certain condition to arise in order to be exploited.

Left unaddressed these issues are highly likely to cause problems with the operation of the contract or lead to a situation which allows the system to be exploited in some way.

## A.1.4 - Critical

Critical issues are directly exploitable bugs or security vulnerabilities.

Left unaddressed these issues are highly likely or guaranteed to cause major problems or potentially a full failure in the operations of the contract.

# Appendix 2 - Disclosure

ConsenSys Diligence (“CD”) typically receives compensation from one or more clients (the “Clients”) for performing the analysis contained in these reports (the “Reports”). The Reports may be distributed through other means, including via ConsenSys publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any Third-Party by virtue of publishing these Reports.

PURPOSE OF REPORTS The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of Solidity code and only the Solidity code we note as being within the scope of our review within this report. The Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond Solidity that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty.

CD makes the Reports available to parties other than the Clients (i.e., “third parties”) -- on its GitHub account (https://github.com/ConsenSys). CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

LINKS TO OTHER WEB SITES FROM THIS WEB SITE You may, through hypertext or other computer links, gain access to web sites operated by persons other than ConsenSys and CD. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that ConsenSys and CD are not responsible for the content or operation of such Web sites, and that ConsenSys and CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or mean that ConsenSys and CD endorses the content on that Web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the Reports. ConsenSys and CD assumes no responsibility for the use of third party software on the Web Site and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

TIMELINESS OF CONTENT The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice. Unless indicated otherwise, by ConsenSys and CD.