



# Source Code Review

Prepared for Metronome • May 2018

V1.3

## 1. Table of Contents

[1. Table of Contents](#)

[2. Executive Summary](#)

[3. Introduction](#)

[4. Summary Of Findings](#)

[5. Findings](#)

[MTN-001 - External Founder can be used to bypass distribution rules](#)

[MTN-002 - FixedMath operations with small numbers return unexpected results](#)

[MTN-003 - Use of deprecated 'var' keyword to declare variables](#)

[MTN-004 - Use of future Solidity compiler versions](#)

[MTN-005 - Functions declared in API reference but not implemented and vice versa](#)

[MTN-006 - Use "emit" keyword to raise Events](#)

[MTN-007 - Remove address overloads on contracts](#)

[6. Disclaimer](#)

## 2. Executive Summary

In March 2018, Metronome engaged [Coinspect](#) to perform a security audit of the Metronome Token (MTN) contracts. The contracts were supplied through a Git repository. After the first audit, two re-audits were done to verify the fixes of the issues reported. On May 30th, an additional source code and post deployment review was completed.

The objective of the audits was to evaluate the security of the smart contracts. During the assessment, Coinspect identified the **2 medium-risk and 5 low-risk issues that were fixed**. The issues identified during the assessment were not exploitable by themselves to steal funds.

### 3. Introduction

The Metronome token is a cryptocurrency that presents several interesting features such as blockchain migrations, daily token auctions and a self-regulating exchange, among others.

The token is implemented in one single file called “monolithic.sol”, this file contains several base contracts which are then used to construct the following main contracts:

- *MTNToken* - The Metronome Token contract, ERC20/ERC827 compatible.
- *AutonomousConverter* - The exchange contract where users may exchange MTN for ETH and vice versa.
- *Auctions* - The contract that manages the initial auction for MTN tokens and the daily token auctions where new MTN tokens are minted.
- *Proceeds* - The contract where all ETH raised from the initial auction and all the following daily auctions are stored. This contract will supply the exchange contract with a percentage of the raised ETH daily.
- *TokenLocker* - A contract supplied to each founder that regulates how and when the supplied tokens can be withdrawn.
- *TokenPorter* - Provides export functionality to migrate tokens to a different blockchain.

A whitebox security audit was conducted on these smart contracts. The present report was completed on May 30th, 2018, by Coinspect. The report includes all issues identified in the audit.

The following checks, related to best practices, were performed:

- Confusion of the different method calling possibilities: `send()`, `transfer()`, and `call.value()`
- Missing error handling of external calls
- Erroneous control flow assumptions after external calls
- The use of push over pull for external calls
- Lack of enforcement of invariants with `assert()` and `require()`
- Rounding errors in integer division
- Fallback functions with higher gas limit than 2300
- Functions and state variables without explicitly visibility
- Missing pragmas to for compiler version
- Race conditions, such as contract Reentrancy
- Transaction front running
- Timestamp dependence
- Integer overflow and underflow
- Code blocks that consume a non-constant amount of gas, that grows over block gas limit.
- Denial of Service attacks
- Suspicious or underhanded code.

## 4. Summary Of Findings

ID	Description	Risk
MTN-001	External Founder can be used to bypass distribution rules	Medium
MTN-002	FixedMath operations with small numbers return unexpected results	Medium
MTN-003	Use of deprecated 'var' keyword to declare variables	Low
MTN-004	Use of future Solidity compiler versions	Low
MTN-005	Functions declared in API reference but not implemented and vice versa	Low
MTN-006	Use "emit" keyword to raise Events	Low
MTN-007	Remove address overloads on contracts	Low

## 5. Findings

### MTN-001 External Founder can be used to bypass distribution rules

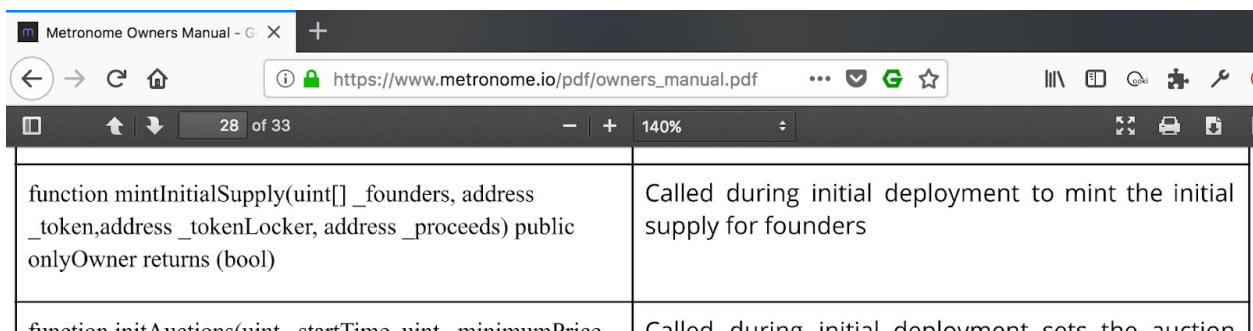
Total Risk <b>Medium</b>	Impact High	Location monolithic.sol:1216
	Likelihood Low	Category Funds Distribution Rules Bypass
	Fixed Yes	<a href="https://github.com/autonomoussoftware/metronome/commit/5fd92332062e38fd064395789c010e4089c52a1d">https://github.com/autonomoussoftware/metronome/commit/5fd92332062e38fd064395789c010e4089c52a1d</a>

#### Description

The MTN Token documentation declares that: “Of the initial 10,000,000 MTN, a one-time authors retention of 2,000,000 MTN will go to Bloq and MTN authors. 25% of this retention will be available at the end of the Initial Supply Auction. The remaining 75% will become available in 12 equal amounts over 12 quarters”. However, using the figure of the External Founder this rule can be circumvented to claim the tokens immediately.

The `mintInitialSupply` function within the Auctions contract takes as an input parameter the address for an External Founder (`_extFounder`). The tokens that were not assigned to any founder within the `_founders` list will be sent to this address, to complete the grand total of 2 million MTN Tokens. If the owner of the contract, which is the responsible of minting the initial supply and starting the auctions phase, decides to assign small number of tokens to the founders then the external founder (which may be himself) will receive a considerably large number of tokens which can be claimed immediately without the time limits initially declared.

The External Founder figure is not mentioned in any of the MTN Token documentation. Furthermore, the [API Reference](#) contains the `mintInitialSupply` function, but omits the use of the `_extFounder`:



<code>function mintInitialSupply(uint[] _founders, address _token, address _tokenLocker, address _proceeds) public onlyOwner returns (bool)</code>	Called during initial deployment to mint the initial supply for founders
<code>function initAuctions(uint _startTime, uint _minimumPrice)</code>	Called during initial deployment sets the auction

## Recommendations

Remove the figure of the External Founder, or make it subject to the same retention distribution rules which are defined in the *TokenLocker* contract instead of transferring the tokens upon minting of the Initial Token Supply.

## MTN-002 FixedMath operations with small numbers return unexpected results

Total Risk <b>Medium</b>	Impact <b>Medium</b>	Location monolithic.sol:80
	Likelihood <b>Low</b>	Category Input Validation
	Fixed <b>No</b>	MTN Team response: "There is 1 call to fSqrt in the code base and the formula that it is used in is: $\text{fSqrt}(1 + E/R)$ so the argument will always be larger than 1."

### Description

The FixedMath contract takes numbers with 18 decimals to perform mathematical operations. If smaller numbers are used, the results returned by the "fMul", "fDiv" and "fSqrt" are not the expected ones. This could be dangerous as these functions are used to calculate the exchange between ETH and MTN Tokens by the "AutonomousConverter" contract.

### Recommendations

Verify the ranges of the input passed to the FixedMath functions.

## MTN-003 Use of deprecated 'var' keyword to declare variables

Total Risk <b>Low</b>	Impact <b>Low</b>	Location monolithic.sol:1051, monolithic.sol:1442, monolithic.sol:1599
	Likelihood <b>Low</b>	Category Deprecated keyword
	Fixed <b>Yes</b>	<a href="https://github.com/autonomoussoftware/metronome/commit/a53b9dcc35d4032717630310836ea773dfd58673">https://github.com/autonomoussoftware/metronome/commit/a53b9dcc35d4032717630310836ea773dfd58673</a>

### Description

As of Solidity version [0.4.20](#), the `var` keyword has been deprecated for security reasons and it will be completely removed as of Solidity 0.5.0. The keyword was found in several lines through the code when assigning values to variables within tuples.

### Recommendations

Declare the variables as a step previous to be assigned in the tuples.

For example, replace:

```
var (time, price, auctionTokens) = nextAuction();
```

with:

```
uint time;  
uint price;  
uint auctionTokens;  
(time, price, auctionTokens) = nextAuction();
```

## MTN-004 Use of future Solidity compiler versions

Total Risk  
**Low**

Impact  
Low

Location  
monolithic.sol:25

Likelihood  
Low

Category  
Solidity Best Practices

Fixed  
Yes

<https://github.com/autonomoussoftware/metronome/commit/5b1303392ca4e3926c361963c138ce758f8bcada>

### Description

According to the [Solidity documentation](#) : “Source files can (and should) be annotated with a so-called version pragma to reject being compiled with future compiler versions that might introduce incompatible changes.”

In the case of the MTN Token contracts the pragma indicates that any compiler version above 0.4.19 should be used, which includes future major releases such as 0.5.0. This contradicts the principle described by the documentation:

```
pragma solidity >=0.4.19;
```

Typically changes in version 0.4.\* will not break the code, but 0.5.0 will probably include new changes that will require the code to be changed in order to compile successfully.

### Recommendations

It is always advisable to use the latest compiler version but not to use a future one without knowing the potential changes that it may introduce, specially if it is a major version release. Add the latest version to the pragma statement:

```
pragma solidity ^0.4.21;
```

The notation used above allows for the use of versions between 0.4.21 and 0.5.0.

### References

For more information on the use of the version pragma and how to handle the different versions accepted, see the following reference links:

- <https://solidity.readthedocs.io/en/develop/layout-of-source-files.html#version-pragma>
- <https://docs.npmjs.com/misc/semver#versions>



## MTN-005 Functions declared in API reference but not implemented and vice versa

Total Risk <b>Low</b>	Impact Low	Location -
	Likelihood Low	Category Unimplemented function
	Fixed Yes	Unimplemented functions removed from Owner Manual Version 0.965 (04.10.2018)

### Description

The [Owners Manual](#) includes an API reference with all publicly available functions (contract internal functions are not included). However, there are several functions that are not implemented in the contracts.

We were able to identify the following missing functions:

- “sendAllSubscriptionPayments” within the MTNToken Contract.
- “balanceEnquiry” within the TokenLocker Contract.
- “startTokenReleaseTimer” within the TokenLocker Contract.
- “initTokenLocker” within the TokenLocker Contract.
- “initTokenPorter” within the TokenPorter Contract.

The last two functions seem to be replaced by the Constructor functions for the contract that contains them.

Additionally, we found that the function “lockTokenLocker” within the TokenLocker Contract was not declared within the API reference.

### Recommendations

Make sure the Contracts public functions and the API reference match.

## MTN-006 Use “emit” keyword to raise Events

Total Risk  
**Low**

Impact  
**Low**

Location  
monolithic.sol

Likelihood  
**Low**

Category  
Solidity Best Practices

Fixed  
Yes

<https://github.com/autonomoussoftware/metronome/commit/5b1303392ca4e3926c361963c138ce758f8bcada>

### Description

Solidity version [0.4.21](#) includes the `emit` keyword which should be used when emitting Events in order to differentiate from function calling. Often this can be confusing as for example ERC-20 tokens have the `transfer` function and the `Transfer` event, which are usually called consecutively:

```
transfer(address to, uint value);  
Transfer(address from, address to, uint256 _value);
```

### Recommendations

Use the `emit` keyword when triggering Events.

### References

Proposal: add `emit` keyword; make it the only way to emit events  
<https://github.com/ethereum/solidity/issues/2877>

## MTN-007 Remove address overloads on contracts

Total Risk  
**Low**

Impact  
**Low**

Likelihood  
**Low**

Fixed  
**Yes**

Location

Monolithic.sol:862, monolithic.sol:973, monolithic.sol:977,  
monolithic.sol:1068, monolithic.sol:1149

Category

Solidity Best Practices

<https://github.com/autonomoussoftware/metronome/commit/2dc73b59a92d33898e3fd0046abea0443b6846ac>

### Description

Solidity version [0.4.21](#) deprecates the use of address properties such as `transfer` and `balance` directly over a contract in order to avoid confusion with similarly named functions for that contract such as `transfer()` and `balanceOf()`. In order to invoke the address properties, the contract instance should be explicitly typecasted to an address.

For example, to transfer ETH to the `autonomousContract` replace:

```
autonomousConverter.transfer(val);
```

With:

```
address(autonomousConverter).transfer(val);
```

### Recommendations

Use `address()` to typecast all contract instances that use the `transfer` or `balance` properties.

### References

Remove address overloads on contracts:

<https://github.com/ethereum/solidity/issues/2683>

Warn on using address members on contracts:

<https://github.com/ethereum/solidity/issues/2695>

A function is called `balance()` confuses compiler with `balance` property:

<https://github.com/ethereum/solidity/issues/2655>

## 6. Disclaimer

The present security audit is limited to smart contract code. It does not cover the technologies and designs related to these smart contracts, nor the frameworks and wallets that communicate with the contracts, nor the general operational security of the company whose contracts have been audited. **This document should not be read as investment advice or an offering of tokens.**